symmetry

MDPI

Article

Incorporating Symmetric Smooth Regularizations into Sparse Logistic Regression for Classification and Feature Extraction

Jing Wang [1,2,*], Xiao Xie [1,2], Pengwei Wang [1,2], Jian Sun [1,2], Yaochen Liu [1,2] and Li Zhang [3]

[1] School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China
[2] Henan Key Laboratory of Analysis and Applications of Education Big Data, Xinyang Normal University, Xinyang 464000, China
[3] School of Early-Childhood Education, Nanjing Xiaozhuang University, Nanjing 211171, China
* Correspondence: wangjing@xynu.edu.cn

Abstract: This paper introduces logistic regression with sparse and smooth regularizations (LR-SS), a novel framework that simultaneously enhances both classification and feature extraction capabilities of standard logistic regression. By incorporating a family of symmetric smoothness constraints into sparse logistic regression, LR-SS uniquely preserves underlying structures inherent in structured data, distinguishing it from existing approaches. Within the minorization–maximization (MM) framework, we develop an efficient optimization algorithm that combines coordinate descent with soft-thresholding techniques. Through extensive experiments on both simulated and real-world datasets, including time series and image data, we demonstrate that LR-SS significantly outperforms conventional sparse logistic regression in classification tasks while providing more interpretable feature extraction. The results highlight LR-SS's ability to leverage sparse and symmetric smooth regularizations for capturing intrinsic data structures, making it particularly valuable for machine learning applications requiring both predictive accuracy and model interpretability.

Keywords: logistic regression; classification; feature extraction; sparse regularization; symmetric smooth regularization; minorization–maximization

Academic Editor: Theodore E. Simos

Received: 20 December 2024
Revised: 16 January 2025
Accepted: 18 January 2025
Published: 21 January 2025

Citation: Wang, J.; Xie, X.; Wang, P.; Sun, J.; Liu, Y.; Zhang, L. Incorporating Symmetric Smooth Regularizations into Sparse Logistic Regression for Classification and Feature Extraction. *Symmetry* **2025**, *17*, 151. https://doi.org/10.3390/sym17020151

1. Introduction

Logistic regression (LR) [1] has long been a cornerstone in binary classification tasks across various domains. Its versatility is evident in its wide-ranging applications, from predicting disease risks and patient mortality rates in medicine [2–4] to forecasting voting behaviors in political science [5], assessing system failures in engineering [6], and identifying key indicators for successful foreign direct investment in finance [7]. The extension of logistic regression to sequential data through conditional random fields (CRFs) [8] has further broadened its utility, particularly in natural language processing.

A key advancement in logistic regression has been the development of sparse logistic regression (SLR), which performs feature selection by enforcing sparsity in model coefficients through L1-norm (lasso) or other non-convex regularizations [9,10]. This approach selects only the most relevant features, making the model more interpretable and resistant to overfitting, which is particularly valuable in high-dimensional settings where predictors outnumber observations [11]. The effectiveness of SLR has been demonstrated across diverse domains, including bioinformatics [12,13] and neuroimaging [14,15], where both feature selection and model interpretability are crucial.

However, traditional sparse logistic regression has a significant limitation: it fails to leverage inherent structural relationships between predictors, particularly in datasets with temporal or spatial dependencies. For example, in EEG-based motor imagery classification, SLR produces predictive maps with discretely activated channels on the scalp [14]. Similarly, in fMRI-based functional task classification, the predictive maps reveal distinct activation regions within brain space [15]. This discrete nature of activation patterns fails to capture the smooth spatial relationships that naturally exist in various datasets.

When predictors exhibit natural ordering or grouping structures, such as in time series biomarkers or spatially distributed signals, incorporating smoothness constraints alongside sparsity can better capture underlying patterns [16,17]. For instance, in medical diagnostics, biomarkers typically show gradual changes over time or space, making smooth variations in model coefficients essential for both prediction accuracy and result interpretability [16]. Moreover, smooth models often demonstrate superior stability and convergence properties. Algorithms designed for smooth approximations of non-differentiable penalties achieve faster convergence and computational efficiency, as evidenced in methods like Lassplore and adaptive line search schemes [18]. The addition of smoothness constraints also enhances model robustness to noise, as demonstrated in applications such as Raman spectral data analysis [17].

Given the compelling advantages of smooth models, researchers have developed numerous methods that combine smooth constraints with sparse regularizations, particularly in brain decoding applications. For instance, Grosenick et al. [19] constructed smooth regularizations based on GraphNet, and de Brecht et al. [20] developed smooth sparse logistic regression (SSLR) by introducing a smooth regularization using the inverse of the adjacency matrix. Building upon these approaches, Watanabe et al. [21] integrated the 6-D structure of the functional connectome into either fused lasso (FL) or GraphNet regularizations. Zhang et al. [22] introduced Euler elastica (EE) regularized logistic regression that overcame the limitation of total variation (TV) regularization that favored piece-wise constant rather than piece-wise smooth images. Additionally, Wen et al. [23] designed regularizations with the group sparse property based on prior structural or functional segmented brain atlases. These approaches aim to fully leverage the classification-relevant information from raw data while ensuring that the extracted features adequately reflect the temporal and spatial structures inherent in the original data.

A fundamental characteristic of these regularization approaches is their inherent symmetry. The smooth regularization matrices employed in these methods are predominantly symmetric, a property that reflects the natural reciprocity in spatial and temporal feature relationships. For instance, in spatiotemporal modeling [16,17], adjacent areas demonstrate reciprocal dependencies, while in neuroimaging applications [19,20], neural pathways typically exhibit mutual influences between regions. The symmetric structure of these regularization matrices thus provides both mathematical rigor and physical interpretability, making them particularly effective for applications where feature relationships are inherently bidirectional.

Building upon previous research, this paper introduces a family of symmetric smooth matrices into traditional sparse logistic regression, leading to a logistic regression with sparse and smooth regularizations (LR-SS) framework. Compared to existing models, the proposed framework offers greater flexibility in characterizing spatial or temporal structures by considering the relationships between both adjacent and non-adjacent features, thereby enabling more comprehensive utilization of structural information. Furthermore, by adjusting the parameters of the symmetric smooth matrices, our model can naturally reduce to several existing models as special cases.

This paper makes three key contributions: (1) We propose a novel LR-SS framework that leverages symmetric smooth matrices to generalize existing algorithms, including LR with GraphNet regularization and SSLR, with these algorithms emerging as special cases through parameter adjustment. (2) We develop an efficient vectorized iterative solution within the minorization–maximization (MM) framework, including simplified solutions specifically designed for Laplacian matrix-based smooth matrices. (3) We provide comprehensive experimental validation using both simulated and real-world datasets, demonstrating the superior capabilities of LR-SS in classification and feature extraction compared to existing logistic regression algorithms.

The paper is organized as follows: Section 2 establishes the theoretical foundation of LR-SS, including the problem formulation, smooth matrix construction, optimization algorithm, and experimental setup. Section 3 presents comprehensive experimental results on both simulated and real-world datasets. Section 4 provides a detailed discussion of the findings and implications. Finally, Section 5 summarizes our conclusions and outlines future research directions.

2. Materials and Methods

Based on the motivation outlined in the introduction, this section presents the theoretical foundation and methodology of our proposed LR-SS framework. We begin by establishing notation and formulating the basic logistic regression problem, then progressively build up to our full LR-SS model through the incorporation of sparse and smooth regularizations. We also detail the construction of different smooth matrices and present an efficient optimization algorithm.

In this study, we adopt the following notational conventions: lowercase letters denote scalars, bold lowercase letters denote column vectors, and bold uppercase letters denote matrices. The L1-norm and L2-norm are denoted by $\|\cdot\|_1$ and $\|\cdot\|_2$, respectively. We use $\text{sign}(\cdot)$ to represent the sign function. The function $\text{diag}(\cdot)$ serves a dual purpose: when applied to a matrix, it extracts the diagonal elements to form a vector; when applied to a vector, it constructs a diagonal matrix by putting the vector elements on the diagonal. $\mathbf{1}_d$ denotes a d-dimensional column vector of ones. For two vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{a} \circ \mathbf{b}$ denotes the Hadamard product, which represents the element-wise product between vectors $\mathbf{a}$ and $\mathbf{b}$. For a vector $\mathbf{x}$, the ith element is denoted as x_i. For a matrix $\mathbf{Y}$, the ith column is denoted as $\mathbf{y}_i$, the element in the ith row and the jth column is denoted as y_{ij}.

2.1. Problem Formulation

To establish a solid theoretical foundation for our LR-SS framework, we systematically develop the mathematical formulation, starting from basic logistic regression and building up to our complete LR-SS model through the incorporation of various regularization terms.

2.1.1. Logistic Regression

Logistic regression (LR) [1] is widely employed for binary classification tasks. Consider a dataset comprising n independent and identically distributed samples, represented as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n] \in \mathbb{R}^{(d-1)\times n}$, with corresponding binary labels denoted by $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T \in \mathbb{R}^n$, where $y_i \in \{0, 1\}$ for $i = 1, 2, ..., n$. Given a weight vector $\mathbf{w} \in \mathbb{R}^{d-1}$ and an intercept term $w_0 \in \mathbb{R}$, the probability that a sample $\mathbf{x}_i$ belongs to the positive class ($y_i = 1$) can be expressed as

$$P(y_i = 1 \mid \mathbf{x}_i, \mathbf{w}, w_0) = \frac{1}{1 + \exp(-(w_0 + \mathbf{w}^T \mathbf{x}_i))}. \quad (1)$$

To eliminate the intercept term w_0, we construct augmented matrices $\mathbf{x}_i \leftarrow [1; \mathbf{x}_i]$ and $\mathbf{w} \leftarrow [w_0; \mathbf{w}]$. This transformation yields

$$P(y_i = 1|\mathbf{x}_i, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}_i), \tag{2}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ represents the sigmoid function. Consequently, the probability that sample $\mathbf{x}_i$ belongs to category y_i can be expressed as

$$P(y_i|\mathbf{w}, \mathbf{x}_i) = \Big(\sigma(\mathbf{w}^T\mathbf{x}_i)\Big)^{y_i}\Big(1 - \sigma(\mathbf{w}^T\mathbf{x}_i)\Big)^{1-y_i}. \tag{3}$$

The joint probability density function is given by

$$P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{n}\Big(\sigma(\mathbf{w}^T\mathbf{x}_i)\Big)^{y_i}\Big(1 - \sigma(\mathbf{w}^T\mathbf{x}_i)\Big)^{1-y_i}. \tag{4}$$

The weight vector $\mathbf{w}$ can be estimated using the maximum likelihood method. Taking the logarithm of the joint probability density yields the optimization problem for logistic regression (LR):

$$\max_{\mathbf{w}} \ln P(\mathbf{y}|\mathbf{X}, \mathbf{w}). \tag{5}$$

When prior knowledge of $\mathbf{w}$ is available, Bayesian theory allows us to estimate $\mathbf{w}$ through posterior probability maximization. The posterior probability of $\mathbf{w}$ given $\mathbf{X}$ and $\mathbf{y}$ can be expressed as

$$P(\mathbf{w}|\mathbf{X}, \mathbf{y}) \propto P(\mathbf{y}|\mathbf{X}, \mathbf{w})P(\mathbf{w}), \tag{6}$$

where $P(\mathbf{w})$ represents the prior probability of $\mathbf{w}$, defined only on the weight coefficients excluding the intercept term w_0.

2.1.2. Logistic Regression with L2-Norm Regularization

Applying a Gaussian prior to the weight vector $\mathbf{w}$, and then taking the logarithm of the posterior probability, yields the optimization problem for logistic regression with L2-norm regularization (LR-L2) [9,24]:

$$\max_{\mathbf{w}}\left\{\ln P(\mathbf{y}|\mathbf{X}, \mathbf{w}) - \frac{\lambda_2}{2}\|\mathbf{w}\|_2^2\right\}, \tag{7}$$

where λ_2 is a non-negative regularization parameter controlling the strength of the Gaussian priors. The addition of L2-norm regularization helps prevent overfitting by imposing smoothness constraints on the model parameters. This regularization approach serves as an important precursor to our more sophisticated smooth regularization schemes. Note that incorporating the L2-norm regularization into the standard linear regression framework will yield the well-established ridge regression formulation, also known as Tikhonov regularization [25].

2.1.3. Logistic Regression with L1-Norm Regularization

Applying a Laplacian prior to the weight vector $\mathbf{w}$, and then taking the logarithm of the posterior probability, yields the optimization problem for logistic regression with L1-norm regularization (LR-L1), also known as sparse logistic regression (SLR) [9,15,26,27]:

$$\max_{\mathbf{w}}\{\ln P(\mathbf{y}|\mathbf{X}, \mathbf{w}) - \lambda_1\|\mathbf{w}\|_1\}, \tag{8}$$

where λ_1 is a non-negative regularization parameter controlling the strength of the Laplacian priors. The incorporation of L1-norm regularization introduces sparsity into the model,

crucial for feature selection and model interpretability. This regularization is also known as lasso regularization [28].

2.1.4. Logistic Regression with ElasticNet Regularization

Having separately examined the Gaussian and Laplacian priors, we now consider applying them to the weight vector $\mathbf{w}$ simultaneously. Then, taking the logarithm of the posterior probability yields the optimization problem for logistic regression with ElasticNet regularization (LR-ElasticNet) [9]:

$$\max_{\mathbf{w}} \left\{ \ln P(\mathbf{y}|\mathbf{X}, \mathbf{w}) - \lambda_1 \|\mathbf{w}\|_1 - \frac{\lambda_2}{2} \|\mathbf{w}\|_2^2 \right\}, \tag{9}$$

where λ_1 and λ_2 are non-negative regularization parameters controlling the strength of the Laplacian and Gaussian priors, respectively. The combination of L1-norm and L2-norm regularizations is known as ElasticNet regularization [29].

2.1.5. Logistic Regression with Sparse and Smooth Regularizations

Replacing the L2-norm regularization in LR-ElasticNet with a smooth regularization yields the optimization problem for logistic regression with sparse and smooth regularizations (LR-SS):

$$\max_{\mathbf{w}} \left\{ \ln P(\mathbf{y}|\mathbf{X}, \mathbf{w}) - \lambda_1 \|\mathbf{w}\|_1 - \frac{\lambda_2}{2} \mathbf{w}^T \mathbf{Q} \mathbf{w} \right\}, \tag{10}$$

where $\mathbf{w}^T \mathbf{Q} \mathbf{w}$ is the smooth regularization, and λ_1 and λ_2 are non-negative regularization parameters controlling the strength of the Laplacian and smooth priors, respectively. This generalization allows for better capture of temporal and spatial relationships while maintaining the desirable sparsity properties.

The proposed LR-SS algorithm is a general framework that encompasses several existing methods, including spatially regularized logistic regression (SRLR) [30], spatially regularized sparse logistic regression (SRSLR) [31], smooth sparse logistic regression (SSLR) [20], and logistic regression with GraphNet regularization [19]. These algorithms are rooted in the graphical lasso theory [32,33], which models a Gaussian prior whose covariance matrix is not an identity matrix. Consequently, the inverse of the covariance matrix, or equivalently the smooth matrix $\mathbf{Q}$ in Equation (10), has non-zero offdiagonal elements that are capable of capturing complex temporal and spatial relationships between features.

The optimization problems of the five algorithms are summarized in Table 1. When $\lambda_1 = \lambda_2 = 0$, LR-SS degenerates to LR. When $\lambda_1 = 0$ and $\mathbf{Q} = \mathbf{I}$, LR-SS degenerates to LR-L2. When $\lambda_2 = 0$, LR-SS degenerates to LR-L1. When $\mathbf{Q} = \mathbf{I}$, LR-SS degenerates to LR-ElasticNet. Therefore, LR-SS is a generalized form of the other four algorithms.

Table 1. Summary of optimization problems for different logistic regression algorithms.

Algorithm	Optimization Problem
LR	$\max_{\mathbf{w}} \ln P(\mathbf{y}\|\mathbf{X}, \mathbf{w})$
LR-L2	$\max_{\mathbf{w}} \left\{ \ln P(\mathbf{y}\|\mathbf{X}, \mathbf{w}) - \frac{\lambda_2}{2} \|\mathbf{w}\|_2^2 \right\}$
LR-L1	$\max_{\mathbf{w}} \{ \ln P(\mathbf{y}\|\mathbf{X}, \mathbf{w}) - \lambda_1 \|\mathbf{w}\|_1 \}$
LR-ElasticNet	$\max_{\mathbf{w}} \left\{ \ln P(\mathbf{y}\|\mathbf{X}, \mathbf{w}) - \lambda_1 \|\mathbf{w}\|_1 - \frac{\lambda_2}{2} \|\mathbf{w}\|_2^2 \right\}$
LR-SS	$\max_{\mathbf{w}} \left\{ \ln P(\mathbf{y}\|\mathbf{X}, \mathbf{w}) - \lambda_1 \|\mathbf{w}\|_1 - \frac{\lambda_2}{2} \mathbf{w}^T \mathbf{Q} \mathbf{w} \right\}$

2.1.6. Classification

Once $\mathbf{w}$ is computed by one of the above LR algorithms, for a given test sample $\mathbf{z}$, the probability that the sample belongs to the positive class can be calculated using the following logistic function:

$$P(y = 1|\mathbf{z}, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{z}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{z})}. \tag{11}$$

This probability can be used to classify the sample by applying an appropriate threshold (typically 0.5). By classifying all test samples and calculating the average classification accuracy, the overall performance of a specific logistic regression algorithm can be assessed.

2.2. Smooth Matrix Construction

Having established the basic framework of LR-SS, we now turn to the crucial task of building appropriate smooth models that can effectively capture the temporal and spatial relationships in the data. The smooth properties can be characterized in various ways [19–23]. This paper focuses on constructing smooth matrices that can be readily incorporated into the LR-SS framework. The different approaches are presented and their properties are analyzed throught visual comparisons.

2.2.1. Smooth Matrix Based on the Laplacian Matrix

Our first approach utilizes the Laplacian matrix, which provides a natural way to capture symmetric relationships between neighboring features in a graph structure. The construction proceeds as follows. Let $\mathbf{a}_i$ and $\mathbf{a}_j, i, j = 1, 2, ..., d$ be the coordinates of any two features in spatial or temporal dimensions. The distance between them is $d_{ij} = \left\|\mathbf{a}_i - \mathbf{a}_j\right\|_2$. Then, the adjacency matrix $\mathbf{N}$ is defined as

$$N_{ij} = \begin{cases} \exp\left(-\frac{d_{ij}^2}{2\delta^2}\right), & 0 < d_{ij} \leq \varepsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

This construction ensures that $\mathbf{N}$ is symmetric, as $d_{ij} = d_{ji}$. After obtaining the adjacency matrix $\mathbf{N}$, we calculate the degree matrix $\mathbf{D} = \text{diag}(\mathbf{1}_d^T\mathbf{N})$. The Laplacian matrix is then defined as $\mathbf{L} = \mathbf{D} - \mathbf{N}$. The Laplacian matrix inherits symmetry from $\mathbf{N}$ and $\mathbf{D}$, and is positive semi-definite [34]. The symmetry property captures bidirectional relationships between features, while positive semi-definiteness ensures the convexity of the regularization term in the optimization problem. We therefore define the smooth matrix by the Laplacian matrix, i.e., $\mathbf{Q} = \mathbf{L}$, and denote this Laplacian-based smooth matrix as $\mathbf{Q}^{(1)}$ to distinguish it from other variants introduced later.

The parameters δ and ε in $\mathbf{N}$ serve as tuning parameters. The parameter δ controls the magnitude of the non-zero elements in $\mathbf{N}$, with smaller δ resulting in smaller non-zero elements. The parameter ε regulates the sparsity of $\mathbf{N}$, with smaller ε leading to a sparser $\mathbf{N}$, reducing storage and computational requirements. When $\varepsilon = 1$, the smooth regularization only considers symmetric relationships between weights of neighboring features, smoothing only local information. When $\varepsilon > 1$, it also considers symmetric relationships between weights of non-adjacent features, potentially achieving a global smooth effect and improving the algorithm's classification and feature extraction capabilities.

In the one-dimensional case, the smooth regularization can be expressed as the following quadratic form [35]:

$$\mathbf{w}^T\mathbf{Q}\mathbf{w} = \mathbf{w}^T\mathbf{L}\mathbf{w} = \frac{1}{2}\sum_{i,j=1}^{d} N_{ij}(w_i - w_j)^2. \tag{13}$$

This symmetric formulation encourages the weights w_i and w_j to be similar when the corresponding features are strongly connected (i.e., when N_{ij} is large), thus promoting smoothness in the weight vector $\mathbf{w}$.

2.2.2. Smooth Matrix Based on GraphNet

A notable special case of the smooth matrix $\mathbf{Q}^{(1)}$ arises when $\varepsilon = 1$, simplifying the adjacency matrix to

$$N_{ij} = \begin{cases} c, & d_{ij} = 1, \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

where $c = \exp\left(-\frac{1}{2\delta^2}\right)$. In this case, the product $c\lambda_2$ serves as the effective regularization parameter for the smooth regularization term in Equation (10). Without loss of generality, we can set $\delta = \infty$ (equivalently, $c = 1$), leaving λ_2 as the sole smooth regularization parameter. This simplification results in a smooth matrix $\mathbf{Q}$ that is independent of δ.

This simplified form, as a key component of GraphNet [19], only considers symmetric relationships between adjacent features, reducing computational complexity while maintaining effective weight smoothing. The resulting symmetric structure has influenced various interpretable graph neural network architectures in neuroscience [36–39].

The GraphNet regularization is closely related to several other existing smooth regularizations. For the one-dimensional case, we can derive an alternative formulation by substituting Equation (14) into Equation (13):

$$\mathbf{w}^T\mathbf{Q}\mathbf{w} = \frac{1}{2}\sum_{i,j=1}^{d} N_{ij}(w_i - w_j)^2 = \sum_{i=1}^{d-1}(w_{i+1} - w_i)^2 = \|\mathbf{P}\mathbf{w}\|_2^2 = \mathbf{w}^T\mathbf{P}^T\mathbf{P}\mathbf{w}, \tag{15}$$

where $\mathbf{P}$ is the first-order difference matrix with elements of -1 and 1 on the bidiagonal:

$$\mathbf{P} = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{(d-1)\times d}. \tag{16}$$

The smooth matrix $\mathbf{Q}$ can be calculated as

$$\mathbf{Q} = \mathbf{P}^T\mathbf{P} = \begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{d\times d}. \tag{17}$$

In this one-dimensional case, matrix $\mathbf{Q}$ is symmetric and tridiagonal, with diagonal elements equal to 2 (except for the first and last elements, which equal 1) and offdiagonal elements equal to -1.

Replacing L2-norm in Equation (15) with L1-norm yields the total variation (TV) regularization, i.e., $\|\mathbf{P}\mathbf{w}\|_1 = \sum_{i=1}^{d-1}|w_{i+1} - w_i|$ [40]. Combining TV regularization with lasso regularization generates the fussed lasso (FL) regularization, i.e., $\lambda_1\|\mathbf{w}\|_1 + \lambda_2\sum_{i=1}^{d-1}|w_{i+1} - w_i|$ [41], where λ_1 and λ_2 control sparsity and smoothness, respectively. The sparse regularization promotes sparsity by shrinking weights to zero, while the TV regularization encourages adjacent weights to be similar, producing piece-wise constant solutions [18,22,42]. While these alternative formulations are noteworthy, our primary focus remains on analyzing

the smooth effects of various symmetric smooth matrices. Therefore, discussing these alternative formulations is beyond the scope of this study.

2.2.3. Smooth Matrix Based on the Inverse of the Adjacency Matrix

Another definition of the smooth matrix $\mathbf{Q}$ is the inverse of the adjacency matrix $\mathbf{N}$, i.e., $\mathbf{Q} = \mathbf{N}^{-1}$ [20]. This smooth matrix, denoted as $\mathbf{Q}^{(2)}$, is defined such that the correlation strength between weights is directly proportional to a distance measure between the weights in feature space. While this construction inherits symmetry from $\mathbf{N}$, it does not guarantee positive semi-definiteness like the Laplacian-based approaches. Nevertheless, by carefully adjusting the parameters δ and ε of the adjacency matrix, this smooth matrix can achieve effective smoothing of the weights [20].

Table 2 summarizes the construction methods for different smooth matrices used in this study. The table presents three main approaches: the Laplacian matrix-based method, the GraphNet-based method, and the inverse matrix-based method. Each approach has its unique construction formula and characteristics. The Laplacian and GraphNet methods both utilize the $\mathbf{Q}^{(1)}$ framework but differ in their adjacency matrix definitions, while the inverse matrix method employs $\mathbf{Q}^{(2)}$ by directly inverting the adjacency matrix. These approaches offer different smoothing properties and may be more suitable for certain types of data structures.

Table 2. Construction methods of different smooth matrices.

Smooth Matrix	Construction Process
$\mathbf{Q}^{(1)}$	1. Calculate $N_{ij} = \begin{cases} \exp\left(-\frac{d_{ij}^2}{2\delta^2}\right), & 0 < d_{ij} \leq \varepsilon \\ 0, & \text{otherwise} \end{cases}$ 2. Form diagonal matrix $\mathbf{D}$ with $D_{ii} = \sum_{j=1}^{d} N_{ij}$ 3. Calculate $\mathbf{Q}^{(1)} = \mathbf{D} - \mathbf{N}$
$\mathbf{Q}^{(1)}$ with GraphNet	1. Calculate $N_{ij} = \begin{cases} 1, & d_{ij} = 1 \\ 0, & \text{otherwise} \end{cases}$ 2. Form diagonal matrix $\mathbf{D}$ with $D_{ii} = \sum_{j=1}^{d} N_{ij}$ 3. Calculate $\mathbf{Q}^{(1)} = \mathbf{D} - \mathbf{N}$
$\mathbf{Q}^{(2)}$	1. Calculate $N_{ij} = \begin{cases} \exp\left(-\frac{d_{ij}^2}{2\delta^2}\right), & 0 < d_{ij} \leq \varepsilon \\ 0, & \text{otherwise} \end{cases}$ 2. Calculate $\mathbf{Q}^{(2)} = \mathbf{N}^{-1}$

2.2.4. Parameter Selection for Smooth Matrices

As outlined above, the smooth matrix $\mathbf{Q}$ is built from an adjacency matrix $\mathbf{N}$, whose entries depend on two key parameters, δ and ε. The construction ensures that the smooth matrix $\mathbf{Q}$ (e.g., a Laplacian matrix or its variant) captures symmetric relationships tied to physical or topological proximity of features.

The parameter δ can be viewed as the bandwidth controlling the degree to which distant features are still regarded as connected. A large δ spreads out the similarity measure so that features far apart in distance still influence each other, leading to more global smoothing. Conversely, a small δ localizes the smoothing, causing more pronounced weight similarity among only those features that are very close. In datasets with smoothly varying signals (e.g., spatially continuous measurements), a larger δ may better preserve global coherence. However, in environments where local continuity is paramount (e.g., signals that change sharply), a smaller δ may reduce over-smoothing.

The parameter ε controls the maximum distance threshold beyond which two features are considered too far apart to exert mutual influence in the adjacency matrix. This parameter has a direct impact on the sparsity of the adjacency matrix $\mathbf{N}$. A small ε leads to a more localized smoothing focus, limiting adjacency to immediate neighbors or nearest neighbors. The resulting $\mathbf{N}$ is sparser, which can reduce the computational cost while focusing on local structure. On the other hand, a large ε includes more distant pair-wise relationships, which can lead to a denser $\mathbf{N}$. This may capture more global structure but could increase both model complexity and computational overhead.

In practice, δ and ε are often tuned together through a systematic approach. A common starting point is to set δ proportionate to a characteristic scale of the data (e.g., median pair-wise distance among features), while ε is initially chosen to include a moderate number of neighbors around each feature. These initial values can then be refined through cross-validation or model selection, performing a grid search or Bayesian optimization over a range of values while assessing classification performance and smoothness of extracted features on a validation set.

2.2.5. Visual Comparison of Smooth Matrices

To better understand the characteristics and differences between the proposed smooth matrices, we provide a visual comparison using a simple one-dimensional example with 11 features. Figure 1 illustrates the inherent symmetry and structural patterns of these matrices through heatmap visualizations, allowing for direct comparison of their properties.

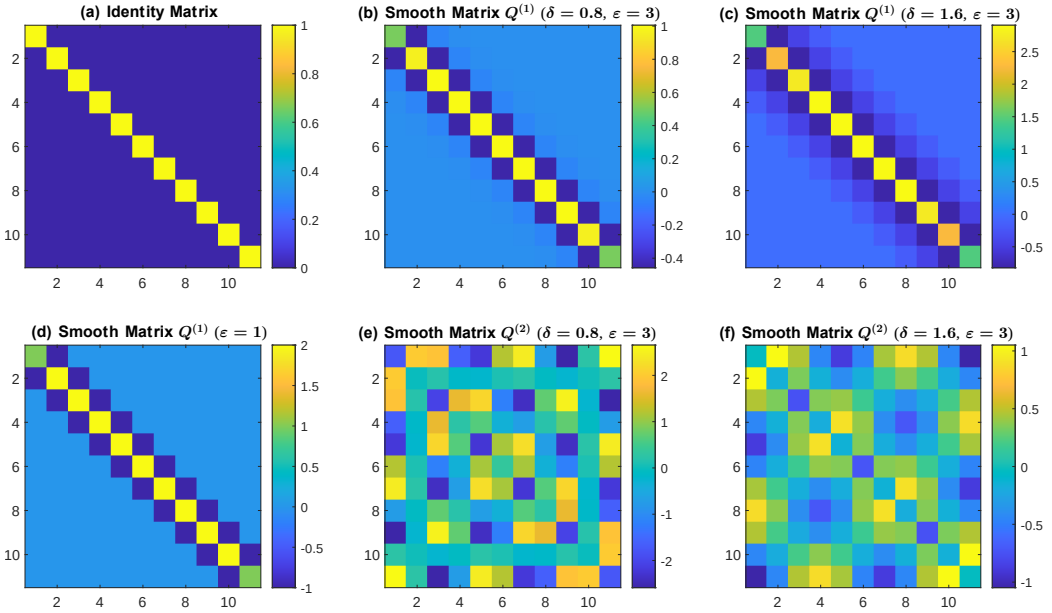

Figure 1. Illustration of smooth matrices. (**a**) Identity matrix. (**b**) $\mathbf{Q}^{(1)}$ with $\delta = 0.8$ and $\varepsilon = 3$. (**c**) $\mathbf{Q}^{(1)}$ with $\delta = 1.6$ and $\varepsilon = 3$. (**d**) $\mathbf{Q}^{(1)}$ with $\varepsilon = 1$. (**e**) $\mathbf{Q}^{(2)}$ with $\delta = 0.8$ and $\varepsilon = 3$. (**f**) $\mathbf{Q}^{(2)}$ with $\delta = 1.6$ and $\varepsilon = 3$.

Figure 1a displays the identity matrix for comparison. Figure 1b shows $\mathbf{Q}^{(1)}$ with $\delta = 0.8$ and $\varepsilon = 3$, where the non-zero elements with large magnitudes are symmetrically distributed in the tridiagonal region of the matrix. The symmetric pattern extends beyond the tridiagonal region with smaller magnitudes, reflecting the bidirectional influence between non-adjacent features in the smoothing process.

Figure 1c shows $\mathbf{Q}^{(1)}$ with $\delta = 1.6$ and $\varepsilon = 3$, where the non-zero elements have larger absolute values compared to Figure 1b, particularly in the central region of the matrix, while maintaining perfect symmetry about the diagonal.

Figure 1d shows $\mathbf{Q}^{(1)}$ with $\varepsilon = 1$, which is equivalent to GraphNet [19]. This special case exhibits a symmetric tridiagonal structure where all elements outside the tridiagonal band are zero, as the adjacency matrix only considers symmetric relationships between direct neighbors.

Figure 1b–d all exhibit symmetric tridiagonal structures, which are crucial for achieving the smooth effect. The symmetric tridiagonal structure ensures that each weight is influenced equally by its immediate neighbors on both sides, leading to a natural and balanced smoothing of the weight values across adjacent features. When $\varepsilon > 1$, symmetric relationships between non-adjacent features are also considered, which may help extract more structural features from the data while maintaining the symmetry that ensures stable optimization.

Figure 1e shows $\mathbf{Q}^{(2)}$ with $\delta = 0.8$ and $\varepsilon = 3$, and Figure 1f shows $\mathbf{Q}^{(2)}$ with $\delta = 1.6$ and $\varepsilon = 3$. Both $\mathbf{Q}^{(2)}$ matrices maintain symmetry as they are derived from the inverse of symmetric adjacency matrices. However, they exhibit a more diffuse pattern without the clear tridiagonal structure seen in the $\mathbf{Q}^{(1)}$ matrices. The lack of concentrated local influence may affect their ability to enforce smoothness between adjacent features.

2.2.6. Special Cases of LR-SS

With the framework and smooth matrices presented above, we now demonstrate how LR-SS serves as a unifying framework that generalizes several existing methods. LR-SS has four key parameters to be tuned: the sparse regularization parameter λ_1, the smooth regularization parameter λ_2, and two parameters δ and ε used for constructing the smooth matrices. By carefully selecting specific parameter values, various well-known algorithms emerge as special cases of LR-SS, which we describe below.

When $\lambda_1 = 0$ and $\lambda_2 = 0$, LR-SS degenerates into standard logistic regression [1], denoted as LR. When $\lambda_1 = 0$, $\lambda_2 \neq 0$ and $\mathbf{Q} = \mathbf{I}$, LR-SS degenerates into logistic regression with L2-norm regularization [1], denoted as LR-L2. When $\lambda_1 \neq 0$ and $\lambda_2 = 0$, LR-SS degenerates into logistic regression with L1-norm regularization, which is the standard sparse logistic regression [9,10], denoted as LR-L1. When $\lambda_1 \neq 0$, $\lambda_2 \neq 0$ and $\mathbf{Q} = \mathbf{I}$, LR-SS degenerates into logistic regression with ElasticNet regularization, denoted as LR-ElasticNet [9]. When $\lambda_1 \neq 0$, $\lambda_2 \neq 0$, $\mathbf{Q} = \mathbf{Q}^{(1)}$ and $\varepsilon = 1$, LR-SS degenerates into logistic regression with GraphNet regularization, denoted as LR-GraphNet [19]. When $\lambda_1 \neq 0$, $\lambda_2 \neq 0$ and $\mathbf{Q} = \mathbf{Q}^{(1)}$, the first form of LR-SS is obtained, denoted as LR-SS1. When $\lambda_1 \neq 0$, $\lambda_2 \neq 0$ and $\mathbf{Q} = \mathbf{Q}^{(2)}$, the second form of LR-SS is obtained, denoted as LR-SS2.

This analysis not only illustrates the versatility of our approach but also positions it within the broader context of regularized logistic regression algorithms. Table 3 summarizes the special cases of LR-SS with different parameter settings. Among these algorithms, LR-GraphNet, LR-SS1, and LR-SS2 incorporate both sparse and smooth regularizations.

Table 3. Special cases of LR-SS with different parameter settings.

Parameter Settings	Algorithm
$\lambda_1 = 0, \lambda_2 = 0$	LR
$\lambda_1 = 0, \lambda_2 \neq 0, \mathbf{Q} = \mathbf{I}$	LR-L2
$\lambda_1 \neq 0, \lambda_2 = 0$	LR-L1
$\lambda_1 \neq 0, \lambda_2 \neq 0, \mathbf{Q} = \mathbf{I}$	LR-ElasticNet
$\lambda_1 \neq 0, \lambda_2 \neq 0, \mathbf{Q} = \mathbf{Q}^{(1)}, \varepsilon = 1$	LR-GraphNet
$\lambda_1 \neq 0, \lambda_2 \neq 0, \mathbf{Q} = \mathbf{Q}^{(1)}$	LR-SS1
$\lambda_1 \neq 0, \lambda_2 \neq 0, \mathbf{Q} = \mathbf{Q}^{(2)}$	LR-SS2

2.2.7. Relationships with Existing Approaches

The proposed LR-SS framework unifies and extends several existing regularized logistic regression approaches [19,20,30,31] through a flexible family of symmetric smooth matrices. LR-GraphNet represents a special case of LR-SS when $\mathbf{Q} = \mathbf{Q}^{(1)}$ and $\varepsilon = 1$, while SSLR [20] can be viewed as LR-SS with $\mathbf{Q} = \mathbf{Q}^{(2)}$, i.e., LR-SS2.

Note that GraphNet regularization [19] was originally designed to enhance traditional linear regression, linear discriminant analysis, support vector machines, and their variants. Its application to logistic regression posed computational challenges due to the nonlinear nature of the logistic function and the presence of both smooth and nonsmooth regularization terms. Our LR-SS framework incorporates GraphNet regularization to logistic regression through careful construction of smooth matrices, resulting in LR-GraphNet.

Several existing methods are closely related to LR-GraphNet and can, therefore, be incorporated into the LR-SS framework. For example, spatially regularized logistic regression (SRLR) [30] corresponds to LR-GraphNet without sparse regularization ($\lambda_1 = 0$). Spatially regularized sparse logistic regression (SRSLR) [31] employs a discrete approximation to the integral of the 3D Laplacian of the weight vector to enforce spatial smoothness, equivalent to applying GraphNet regularization in 3D space.

While LR-SS extends these existing approaches, it offers significant advantages over them. Unlike traditional sparse logistic regression, which treats features independently, LR-SS explicitly models feature relationships through the smooth matrix $\mathbf{Q}$, making it particularly valuable for datasets with complex temporal and spatial dependencies. The framework provides greater flexibility in modeling different types of smooth structures through the parameters δ and ε, allowing it to adapt to varying degrees of smoothness and capture both local and global dependencies. This contrasts with GraphNet's limitation to local neighborhood structures or SSLR's fixed smoothing effect determined by the inverse adjacency matrix.

Furthermore, the unified framework enables systematic comparison and analysis of different regularization approaches, providing insights into their relative strengths and limitations. Through careful parameter selection and smooth matrix construction, LR-SS can be tailored to specific application requirements while maintaining the computational efficiency and theoretical guarantees of its special cases.

2.3. Iterative Solutions

Having fully specified the LR-SS framework and different smooth matrices, we now focus on solving the resulting optimization problem. We first present the minorization–maximization (MM) framework, which provides an elegant approach to handle both the smooth and nonsmooth components of our objective function. Under this framework, we derive an element-wise iterative solution using coordinate descent and soft-thresholding techniques, which we then extend to a more efficient vectorized form. We also analyze two special cases where the smooth matrix Q is either the identity matrix or a Laplacian matrix, leading to simplified solutions. Finally, we present a complete algorithm procedure that encompasses all these scenarios.

2.3.1. Minorization–Maximization Framework

The optimization problem of LR-SS contains nonsmooth terms from both the logistic regression formulation and L1-norm regularization. To solve this challenging problem, we employ the minorization–maximization (MM) framework [43,44], which solves a nons-

mooth optimization problem by iteratively optimizing a simpler surrogate function. The surrogate function must satisfy two conditions:

$$\begin{aligned} f(\mathbf{w}^{(k)}) &= g(\mathbf{w}^{(k)}|\mathbf{w}^{(k)}), \\ f(\mathbf{w}) &\geq g(\mathbf{w}|\mathbf{w}^{(k)}), \quad \forall \mathbf{w}, \end{aligned} \tag{18}$$

where $f(\mathbf{w})$ is the original objective function to be maximized, $g(\mathbf{w}|\mathbf{w}^{(k)})$ is the surrogate function, and $\mathbf{w}^{(k)}$ is the weight vector at the kth iteration. The first equation represents the tangency condition, while the second represents the minorization condition. The MM algorithm proceeds by iteratively maximizing the surrogate function:

$$\mathbf{w}^{(k+1)} = \arg\max_{\mathbf{w}} g(\mathbf{w}|\mathbf{w}^{(k)}). \tag{19}$$

This process guarantees monotonic improvement:

$$f(\mathbf{w}^{(k+1)}) \geq g(\mathbf{w}^{(k+1)}|\mathbf{w}^{(k)}) \geq g(\mathbf{w}^{(k)}|\mathbf{w}^{(k)}) = f(\mathbf{w}^{(k)}). \tag{20}$$

The first inequality follows from the minorization condition, while the second inequality results from the maximization step. This sequence ensures that the objective function value increases with each iteration until convergence to a local optimum.

2.3.2. Element-Wise Iterative Solution

Building upon the MM framework, we now derive an iterative solution algorithm for the LR-SS optimization problem. Let $l(\mathbf{w}) = \ln P(\mathbf{y}|\mathbf{X}, \mathbf{w})$; then, the LR-SS optimization problem can be expressed as

$$f(\mathbf{w}) = l(\mathbf{w}) - \lambda_1 \|\mathbf{w}\|_1 - \frac{\lambda_2}{2}\mathbf{w}^T\mathbf{Q}\mathbf{w}. \tag{21}$$

Performing a second-order Taylor expansion on $l(\mathbf{w})$, and by the mean value theorem, there exists $\theta \in [0, 1]$ such that

$$\begin{aligned} l(\mathbf{w}) =& l(\mathbf{w}^{(k)}) + (\mathbf{w} - \mathbf{w}^{(k)})^T \frac{\partial l(\mathbf{w}^{(k)})}{\partial \mathbf{w}} \\ &+ \frac{1}{2}(\mathbf{w} - \mathbf{w}^{(k)})^T \frac{\partial^2 l\left(\theta\mathbf{w} + (1-\theta)\mathbf{w}^{(k)}\right)}{\partial \mathbf{w}\partial \mathbf{w}^T}(\mathbf{w} - \mathbf{w}^{(k)}). \end{aligned} \tag{22}$$

Define

$$\mathbf{s} = [\sigma(y_1\mathbf{w}^T\mathbf{x}_1), \sigma(y_2\mathbf{w}^T\mathbf{x}_2), ..., \sigma(y_n\mathbf{w}^T\mathbf{x}_n)]^T = \sigma\Big(\mathbf{y} \circ (\mathbf{X}^T\mathbf{w})\Big), \tag{23}$$

where $\sigma(\cdot)$ is the element-wise sigmoid function, and $\mathbf{y} \circ (\mathbf{X}^T\mathbf{w})$ denotes the Hadamard product, i.e., the element-wise product between vectors $\mathbf{y}$ and $\mathbf{X}^T\mathbf{w}$. The gradient and Hessian matrix of $l(\mathbf{w})$ can be derived as follows:

$$\mathbf{g}(\mathbf{w}) = \frac{\partial l(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{X}(\mathbf{y} - \mathbf{s}), \tag{24}$$

$$\mathbf{H}(\mathbf{w}) = \frac{\partial^2 l(\mathbf{w})}{\partial \mathbf{w}\partial \mathbf{w}^T} = \mathbf{X}\,\mathrm{diag}(-(\mathbf{1}_n - \mathbf{s}) \circ \mathbf{s})\mathbf{X}^T \geq -\frac{1}{4}\mathbf{X}\mathbf{X}^T, \forall \mathbf{w}. \tag{25}$$

Define $\mathbf{A} = -\frac{1}{4}\mathbf{X}\mathbf{X}^T$; then, we have

$$l(\mathbf{w}) \geq l(\mathbf{w}^{(k)}) + (\mathbf{w} - \mathbf{w}^{(k)})^T\mathbf{g}(\mathbf{w}^{(k)}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^{(k)})^T\mathbf{A}(\mathbf{w} - \mathbf{w}^{(k)}). \tag{26}$$

Construct the surrogate function:

$$\begin{aligned} g(\mathbf{w}|\mathbf{w}^{(k)}) =& l(\mathbf{w}^{(k)}) + (\mathbf{w}-\mathbf{w}^{(k)})^T\mathbf{g}(\mathbf{w}^{(k)}) + \frac{1}{2}(\mathbf{w}-\mathbf{w}^{(k)})^T\mathbf{A}(\mathbf{w}-\mathbf{w}^{(k)}) \\ & - \lambda_1\|\mathbf{w}\|_1 - \frac{\lambda_2}{2}\mathbf{w}^T\mathbf{Q}\mathbf{w}. \end{aligned} \tag{27}$$

This function satisfies the two conditions of the MM framework, i.e., Equation (18), thus being a reasonable surrogate function for $f(\mathbf{w})$. Removing terms unrelated to $\mathbf{w}$ in $g(\mathbf{w}|\mathbf{w}^{(k)})$ gives

$$\hat{g}(\mathbf{w}|\mathbf{w}^{(k)}) = \frac{1}{2}\mathbf{w}^T(\mathbf{A}-\lambda_2\mathbf{Q})\mathbf{w} + \mathbf{w}^T\left(\mathbf{g}(\mathbf{w}^{(k)}) - \mathbf{A}\mathbf{w}^{(k)}\right) - \lambda_1\|\mathbf{w}\|_1. \tag{28}$$

Consequently, one can iteratively maximize $\hat{g}(\mathbf{w}|\mathbf{w}^{(k)})$ to achieve the maximization of $f(\mathbf{w})$.

The maximization of $\hat{g}(\mathbf{w}|\mathbf{w}^{(k)})$ cannot be directly achieved through conventional approaches, owing to its composite structure containing a non-differentiable L1-norm regularization. However, it can be solved efficiently by combining coordinate descent [45,46] and soft-thresholding [47] techniques. Let $\mathbf{B} = -\mathbf{A} + \lambda_2\mathbf{Q} = \frac{1}{4}\mathbf{X}\mathbf{X}^T + \lambda_2\mathbf{Q}$, $\mathbf{c} = \mathbf{g}(\mathbf{w}^{(k)}) - \mathbf{A}\mathbf{w}^{(k)}$, and $\mathbf{g} = \mathbf{g}(\mathbf{w}^{(k)})$, where $-\mathbf{B}$ and $\mathbf{c}$ are the Hessian matrix and the gradient of $\hat{g}(\mathbf{w}|\mathbf{w}^{(k)})$, respectively. Matrix $\mathbf{B}$ is a constant matrix that is independent of $\mathbf{w}$. Vectors $\mathbf{c}$ and $\mathbf{g}$ are functions of $\mathbf{w}^{(k)}$ and are also independent of $\mathbf{w}$. The surrogate function can be rewritten as

$$\hat{g}(\mathbf{w}|\mathbf{w}^{(k)}) = -\frac{1}{2}\mathbf{w}^T\mathbf{B}\mathbf{w} + \mathbf{w}^T\mathbf{c} - \lambda_1\|\mathbf{w}\|_1. \tag{29}$$

When $\mathbf{Q}$ is constructed using the Laplacian matrix or GraphNet, which are symmetric and positive semi-definite matrices [34], the matrix $\mathbf{B}$ inherits these properties and is guaranteed to be symmetric positive semi-definite. These properties are crucial as they ensure the convexity of the optimization problem and guarantee convergence of the iterative algorithm. In contrast, when $\mathbf{Q}$ is constructed using the inverse of the adjacency matrix [20], while symmetry is preserved, positive semi-definiteness of $\mathbf{B}$ is not guaranteed. This approach, therefore, lacks theoretical justification, and we must ensure that λ_2 is sufficiently small to maintain the positive semi-definiteness of $\mathbf{B}$.

The following derivation focuses solely on cases where the smooth matrix is constructed using the Laplacian matrix or GraphNet. For other types of smooth matrix, including the inverse of the adjacency matrix, due to a potential lack of theoretical justification, we simply apply the iterative solution derived from the former case for numerical computation and evaluation.

Without loss of generality, we assume that $\mathbf{B}$ is positive definite, meaning all of the diagonal elements of $\mathbf{B}$ are strictly positive. While this assumption simplifies our analysis and guarantees convergence of the iterative algorithm, it can be relaxed in practice. In cases where some diagonal elements are zeros (which primarily occurs when $\lambda_2 = 0$), we can add a small positive constant ϵ to ensure numerical stability and avoid division by zero. This modification preserves the essential properties of our approach while making it more robust for practical implementations.

Using coordinate descent [45,46], we fix all elements in $\mathbf{w}$ except the ith element w_i. Expanding Equation (29) as a function of w_i and ignoring unrelated terms yields

$$\begin{aligned} & -\frac{1}{2}b_{ii}w_i^2 - \sum_{j=1,j\neq i}^{d} w_j b_{ij} w_i + c_i w_i - \lambda_1|w_i| \\ =& -\frac{1}{2}b_{ii}\left(w_i - \frac{-\sum_{j=1,j\neq i}^{d} w_j b_{ij} + c_i}{b_{ii}}\right)^2 - \lambda_1|w_i| + \frac{\left(-\sum_{j=1,j\neq i}^{d} w_j b_{ij} + c_i\right)^2}{2b_{ii}}, \end{aligned} \tag{30}$$

where b_{ii} is the ith diagonal element of matrix $\mathbf{B}$, and c_i is the ith element of vector $\mathbf{c}$. The soft-thresholding [47] solution to this problem is

$$w_i^{(k+1)} = \text{soft}\left(\frac{-\sum_{j=1,j\neq i}^{d} w_j b_{ij} + c_i}{b_{ii}}, \frac{\lambda_1}{b_{ii}}\right) = \text{soft}\left(w_i + \frac{(-\mathbf{B}\mathbf{w} + \mathbf{c})_i}{b_{ii}}, \frac{\lambda_1}{b_{ii}}\right), \tag{31}$$

where $\text{soft}(\mathbf{a}, \lambda) = (|\mathbf{a}| - \lambda)_+ \text{sign}(\mathbf{a})$ is the soft-thresholding operator. Note that we use the following vector $\mathbf{w}$ to compute $w_i^{(k+1)}$:

$$\mathbf{w} = \left[w_1^{(k+1)}, w_2^{(k+1)}, ..., w_{i-1}^{(k+1)}, w_i^{(k)}, w_{i+1}^{(k)}, ..., w_d^{(k)}\right]^T. \tag{32}$$

That is, the first $i-1$ elements have been updated to the $(k+1)$th iteration, while the last $d-i+1$ elements are still at the kth iteration. To avoid unnecessary confusion, the iteration count of w_i is omitted by default. Iteratively solving for $w_i^{(k+1)}$ by Equation (31) until convergence yields the solution to the LR-SS problem.

2.3.3. Vectorized Iterative Solution

To improve computational efficiency, we now develop a vectorized version of the iterative solution. The previous approach in Equation (31) updates elements of the weight vector sequentially, which can be computationally intensive for high-dimensional problems. By reformulating the solution to enable parallel updates of all elements in the weight vector simultaneously, we can significantly reduce computational overhead and accelerate convergence.

Let us begin by substituting the definitions of matrix $\mathbf{B}$ and vector $\mathbf{c}$ into Equation (31):

$$w_i^{(k+1)} = \text{soft}\left(w_i + \frac{\left(-(-\mathbf{A} + \lambda_2\mathbf{Q})\mathbf{w} + \mathbf{g}(\mathbf{w}^{(k)}) - \mathbf{A}\mathbf{w}^{(k)}\right)_i}{b_{ii}}, \frac{\lambda_1}{b_{ii}}\right), \tag{33}$$

where $\mathbf{w}^{(k)}$ denotes the weight vector $\mathbf{w}$ with all elements updating to the kth iteration, i.e., $\mathbf{w}^{(k)} = [w_1^{(k)}, w_2^{(k)}, \ldots, w_d^{(k)}]^{\mathrm{T}}$. To accelerate the convergence speed of iteration, after calculating each element of $\mathbf{w}$ using coordinate descent, we can instantly update all related quantities, including vectors $\mathbf{s}$, $\mathbf{c}$, and $\mathbf{g}$. Consequently, we can replace $\mathbf{w}^{(k)}$ with $\mathbf{w}$ in Equation (33), yielding

$$\begin{aligned} w_i^{(k+1)} &= \text{soft}\left(w_i + \frac{(-(-\mathbf{A} + \lambda_2\mathbf{Q})\mathbf{w} + \mathbf{g} - \mathbf{A}\mathbf{w})_i}{b_{ii}}, \frac{\lambda_1}{b_{ii}}\right) \\ &= \text{soft}\left(w_i + \frac{(-\lambda_2\mathbf{Q}\mathbf{w} + \mathbf{g})_i}{b_{ii}}, \frac{\lambda_1}{b_{ii}}\right) \\ &= \frac{\text{soft}(b_{ii}w_i - \lambda_2(\mathbf{Q}\mathbf{w})_i + g_i, \lambda_1)}{b_{ii}} \\ &= \frac{\text{soft}((-a_{ii} + \lambda_2 q_{ii})w_i - \lambda_2(\mathbf{Q}\mathbf{w})_i + g_i, \lambda_1)}{-a_{ii} + \lambda_2 q_{ii}}, \end{aligned} \tag{34}$$

where a_{ii} is the ith diagonal element of matrix $\mathbf{A}$, q_{ii} is the ith diagonal element of matrix $\mathbf{Q}$, g_i is the ith element of vector $\mathbf{g}$, and $\mathbf{w}$ contains results from both the kth and $(k+1)$th iterations, as indicated by Equation (32). Strictly following the coordinate descent approach requires updating $\mathbf{w}^{(k+1)}$ element by element. Fortunately, it can be vectorized as follows:

$$\mathbf{w}^{(k+1)} = \frac{\text{soft}((-\mathbf{a} + \lambda_2\mathbf{q}) \circ \mathbf{w} - \lambda_2\mathbf{Q}\mathbf{w} + \mathbf{g}, \lambda_1)}{-\mathbf{a} + \lambda_2\mathbf{q}}, \tag{35}$$

where $\mathbf{a} = \text{diag}(\mathbf{A})$, $\mathbf{q} = \text{diag}(\mathbf{Q})$, and the division of two vectors is conducted in the element-wise manner. The vector $\mathbf{a}$ can be efficiently calculated by $\mathbf{a} = -\frac{1}{4}(\mathbf{X} \circ \mathbf{X})\mathbf{1}_n$. The update rule in Equation (35) can update all elements in the weight vector simultaneously. Therefore, it can be reformulated by replacing $\mathbf{w}$ with $\mathbf{w}^{(k)}$, yielding

$$\mathbf{w}^{(k+1)} = \frac{\text{soft}((-\mathbf{a} + \lambda_2\mathbf{q}) \circ \mathbf{w}^{(k)} - \lambda_2\mathbf{Q}\mathbf{w}^{(k)} + \mathbf{g}, \lambda_1)}{-\mathbf{a} + \lambda_2\mathbf{q}}. \tag{36}$$

This vectorized form efficiently facilitates the update of $\mathbf{w}^{(k+1)}$. Through successive iterations, the algorithm converges to a stationary point that solves the LR-SS optimization problem.

The vectorized update rule can be further simplified when $\mathbf{Q}$ is defined as an identity matrix or Laplacian matrix. For the identity matrix case, we have $\mathbf{Q} = \mathbf{I}$ and $\mathbf{q} = \mathbf{1}_d$. The update rule can be reformulated as

$$\mathbf{w}^{(k+1)} = \frac{\text{soft}(-\mathbf{a} \circ \mathbf{w}^{(k)} + \mathbf{g}, \lambda_1)}{-\mathbf{a} + \lambda_2}. \tag{37}$$

This simple update rule can be utilized to solve LR, LR-L2, LR-L1, and LR-ElasticNet, depending on the regularization parameters λ_1 and λ_2.

For the Laplacian matrix case, we have $\mathbf{q} = \text{diag}(\mathbf{Q}) = \mathbf{N}^T\mathbf{1}_d$. The update rule can be reformulated as

$$\mathbf{w}^{(k+1)} = \frac{\text{soft}\left(-\mathbf{a} \circ \mathbf{w}^{(k)} + \lambda_2\mathbf{N}\mathbf{w}^{(k)} + \mathbf{g}, \lambda_1\right)}{-\mathbf{a} + \lambda_2\mathbf{N}^T\mathbf{1}_d}. \tag{38}$$

The update rule is exclusively dependent on the adjacency matrix $\mathbf{N}$, without requiring the smooth matrix $\mathbf{Q}$. This independence eliminates intermediate computational steps, thereby enhancing computational efficiency. This update rule can be utilized to solve LR-GraphNet or LR-SS1, depending on the parameters δ and ε.

When $\mathbf{Q}$ is neither an identity matrix nor a Laplacian matrix, the LR-SS optimization problem can be solved through the original update rule in Equation (36). In this case, both the smooth matrix $\mathbf{Q}$ and its diagonal vector $\mathbf{q}$ need to be explicitly computed and stored for the iterative updates. Algorithm 1 presents the algorithm procedure for LR-SS.

2.3.4. Computational Complexity Analysis

We now analyze the computational complexity of different solutions for LR-SS, focusing on the element-wise iterative solution, the vectorized iterative solution, and special cases with simplified smooth matrices.

The element-wise iteration solution in Equation (31) represents a straightforward implementation approach. For each iteration, computing $\mathbf{Bw}$ requires $O(d^2)$ operations for each coordinate update, and with d coordinates to update, this leads to $O(d^3)$ operations for the regularization term. Additionally, computing the gradient term requires $O(nd)$ operations, resulting in a total complexity of $O(d^3 + nd)$ per iteration. By caching intermediate results of $\mathbf{Bw}$ computations and updating only the changed coordinates, the computational complexity can be reduced to $O(d^2 + nd)$ per iteration. While this approach offers implementation flexibility, its coordinate-wise update nature still makes parallelization challenging.

The vectorized iterative solution in Equation (36) takes a different strategy, requiring one $\mathbf{Q}\mathbf{w}^{(k)}$ multiplication and one gradient computation per iteration. This leads to an $O(d^2 + nd)$ complexity. This approach enables simultaneous coordinate updates and is more amenable to parallelization and hardware acceleration.

Algorithm 1 Algorithm procedures of LR-SS.

Input: Training data $\mathbf{X}$, labels $\mathbf{y}$, parameters λ_1, λ_2
(and δ, ε if $\mathbf{Q}$ is not identity matrix)
Output: Weight vector $\mathbf{w}$

Calculate vector $\mathbf{a} = -\frac{1}{4}(\mathbf{X} \circ \mathbf{X})\mathbf{1}_n$
Set relative error tolerance $\epsilon = 10^{-3}$ and maximum iterations $k_{\max} = 10^3$
Initialize $k = 0$, error $= 1$, and $\mathbf{w}^{(0)} = \mathbf{1}_d$
while error $> \epsilon$ and $k < k_{\max}$ **do**
  Calculate $\mathbf{s} = \frac{1}{1+\exp(-\mathbf{X}^T\mathbf{w}^{(k)})}$
  Calculate $\mathbf{g} = \mathbf{X}(\mathbf{y} - \mathbf{s})$
  Update $\mathbf{w}^{(k+1)}$ based on $\mathbf{Q}$ type:
  if $\mathbf{Q}$ is identity matrix:
    $\mathbf{w}^{(k+1)} = \frac{\text{soft}(-\mathbf{a}\circ\mathbf{w}^{(k)}+\mathbf{g},\lambda_1)}{-\mathbf{a}+\lambda_2}$
  else if $\mathbf{Q}$ is GraphNet or Laplacian matrix:
    Calculate adjacency matrix $\mathbf{N}$ using δ and ε
    $\mathbf{w}^{(k+1)} = \frac{\text{soft}(-\mathbf{a}\circ\mathbf{w}^{(k)}+\lambda_2\mathbf{N}\mathbf{w}^{(k)}+\mathbf{g},\lambda_1)}{-\mathbf{a}+\lambda_2\mathbf{N}^T\mathbf{1}_d}$
  else if $\mathbf{Q}$ is other types of smooth matrix:
    Calculate $\mathbf{q} = \text{diag}(\mathbf{Q})$
    $\mathbf{w}^{(k+1)} = \frac{\text{soft}((-\mathbf{a}+\lambda_2\mathbf{q})\circ\mathbf{w}^{(k)}-\lambda_2\mathbf{Q}\mathbf{w}^{(k)}+\mathbf{g},\lambda_1)}{-\mathbf{a}+\lambda_2\mathbf{q}}$
  end if
  Calculate error $= \|\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)}\| / \|\mathbf{w}^{(k)}\|$
  $k \leftarrow k + 1$
end while

Special cases of the vectorized iterative solution arise when $\mathbf{Q}$ takes specific forms, as presented in Equations (37) and (38). When $\mathbf{Q}$ is the identity matrix, the $O(d^2)$ matrix multiplication is eliminated entirely, resulting in a total computational complexity of $O(nd)$ per iteration. For sparse Laplacian or GraphNet structures, let s denote the number of non-zero elements in the smooth matrix. The computation of $\mathbf{Q}\mathbf{w}^{(k)}$ reduces to $O(s)$. In these cases, the main computational burden comes from the $O(nd)$ gradient calculation, while the iteration updates require $O(s)$ operations. Therefore, the total computational complexity is $O(nd + s)$ per iteration for sparse cases, which is significantly more efficient than the general case complexity of $O(d^2 + nd)$ when $s \ll d^2$.

In general, the vectorized iterative solution is more efficient than the element-wise iterative solution, making it the preferred choice in most cases. The vectorized approach also facilitates better utilization of modern hardware architectures and parallel computing capabilities, particularly when dealing with large-scale problems. For special cases where the smooth matrix is an identity matrix or a Laplacian matrix, we can leverage simplified solutions to further improve computational efficiency.

2.4. Experimental Setup

This section outlines the experimental setup used to evaluate the LR-SS algorithm. We begin by describing both simulated and real-world datasets that serve as benchmarks for our evaluation. The simulated datasets are specifically designed to test the algorithm's ability to handle sparse and smooth features, while the real-world datasets represent practical applications across different domains. We then detail our parameter selection strategy, including the ranges explored for four key parameters. Finally, we present a comprehensive set of evaluation metrics chosen to assess both the classification and feature extraction performance of the algorithm, enabling a thorough comparison with existing methods. To facilitate reproducibility and further improvements by other researchers, we

have made all source code, datasets, and experimental configurations used in this study publicly available at https://github.com/yuzhounh/LR-SS (released on 17 January 2025).

2.4.1. Simulated Datasets

To assess the performance of LR-SS in classification and feature extraction, we first conducted experiments on simulated datasets, following an approach similar to [20]. The data generation process is stated as follows. For class 0, we randomly generated 200 independent time points from a standard Gaussian distribution with a mean of 0 and a variance of 1. For class 1, we generated another set of Gaussian noise samples and superimposed a sinusoidal signal with an amplitude of 1/2 between time points 80 and 120. For each class, we generated 1000 samples, resulting in a total dataset of 2000 samples. Figure 2 illustrates an example of these two sample classes and the sinusoidal signal.

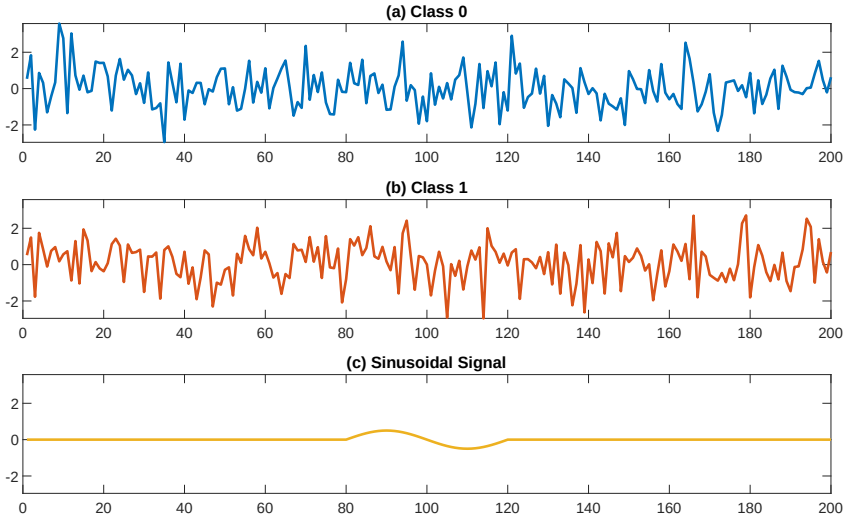

Figure 2. Simulated data showing two classes and the sinusoidal signal. (**a**) Class 0 (blue) consists of pure Gaussian noise. (**b**) Class 1 (red) consists of Gaussian noise superimposed with the sinusoidal signal. (**c**) The sinusoidal signal. The sinusoidal signal is present with amplitude 1/2 between sample points 80 and 120.

This dataset design presents a clear classification challenge: class 0 samples consist purely of random noise, while class 1 samples contain a structured sinusoidal signal embedded within noise. The objective is twofold: to accurately distinguish between these two classes and to extract the features of the embedded sinusoidal signal. The embedded signal introduces a sparse and smooth temporal structure, which is precisely the characteristic that LR-SS is designed to handle through its dual regularization approach. Therefore, this dataset is particularly suitable for validating LR-SS.

2.4.2. Real-World Datasets

Next, we introduce four real-world datasets to evaluate the LR-SS algorithm. The first two datasets are time series data containing a temporal structure: the DistalPhalanxOutlineCorrect database [48,49] for bone outline detection and the GunPoint database [50] for motion classification. The latter two are image datasets containing two-dimensional spatial structures: the FashionMNIST database [51] for fashion item classification and the MNIST database [52] for handwritten digit classification. These diverse datasets allow us to thoroughly evaluate how the proposed algorithm handles both temporal and spatial structures compared to related algorithms.

The DistalPhalanxOutlineCorrect database [48,49] is derived from hand bone X-ray images. It contains data from automated outline detection of the distal phalanx bone,

with human evaluators labeling the outlines as correct or incorrect. The database includes 600 training samples and 276 test samples, with 80 features for each sample. This database can be downloaded from https://www.timeseriesclassification.com/description.php?Dataset=DistalPhalanxOutlineCorrect (accessed on 25 November 2024).

The GunPoint database [50] consists of hand motion tracking data from two actors performing either a gun-drawing or pointing motion. The dataset contains 50 training samples and 150 test samples, with 150 features representing the X-axis motion trajectory. This database can be downloaded from https://timeseriesclassification.com/description.php?Dataset=GunPoint (accessed on 25 November 2024).

The FashionMNIST database [51] contains grayscale images of fashion items. For our binary classification experiments, we only used items labeled as 0 or 1, resulting in 12,000 training samples and 2000 test samples. Each image is 28 × 28 pixels, giving 784 features. This database can be downloaded from https://github.com/zalandoresearch/fashion-mnist (accessed on 4 December 2024).

The MNIST database [52] consists of handwritten digit images. Similar to FashionMNIST, we only used digits 0 and 1 for binary classification, with 12,000 training samples and 2000 test samples. Each image is also 28 × 28 pixels with 784 features. This database can be downloaded from https://yann.lecun.com/exdb/mnist/ (accessed on 4 December 2024).

For the DistalPhalanxOutlineCorrect and GunPoint databases, we retained their original training and test set splits to maintain consistency with prior research. For the FashionMNIST and MNIST datasets, we implemented a computational reduction strategy due to their large sample sizes and associated high computational demands. Specifically, we first combined the original training and test sets, then applied stratified sampling to extract approximately 1000 samples for training while preserving the remaining samples for testing. This sampling approach ensures that the class proportions remain consistent between the training and test sets while significantly reducing computational requirements.

Table 4 provides a detailed overview of the sample sizes and feature dimensions for all four databases, while Figure 3 illustrates representative samples from each dataset.

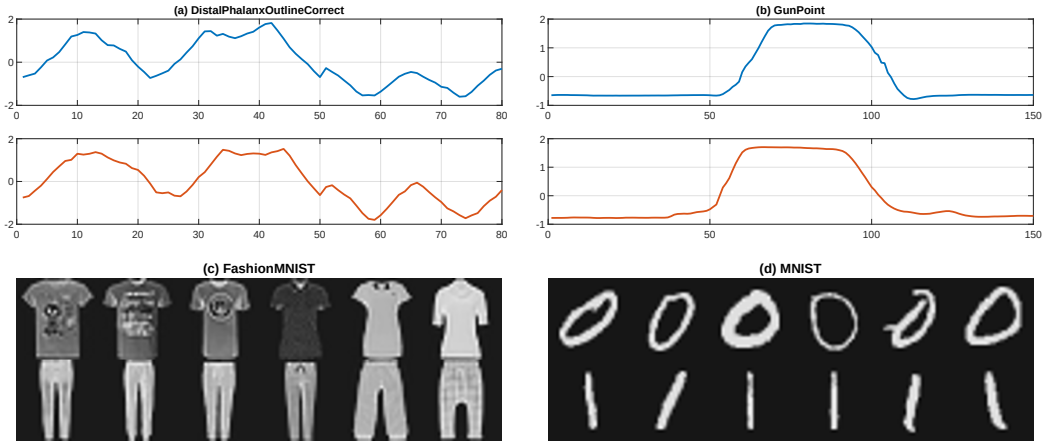

Figure 3. Examples from the four real-world datasets: (**a**) DistalPhalanxOutlineCorrect dataset showing one sample from each class (correct outline in blue vs. incorrect outline in red), (**b**) GunPoint dataset showing one trajectory from each class (gun-drawing in blue vs. pointing in red), (**c**) FashionMNIST dataset showing six samples from each class (T-shirt/top vs. trouser), and (**d**) MNIST dataset showing six samples from each class (digits 0 vs. 1).

Table 4. Summary of database information.

Database	Training Size		Test Size		Features
	Class 0/1	Total	Class 0/1	Total	
DistalPhalanxOutlineCorrect	222/378	600	115/161	276	80
GunPoint	26/24	50	74/76	150	150
FashionMNIST (0–1)	500/500	1000	6500/6500	13000	784
MNIST (0–1)	468/533	1001	6435/7344	13779	784

2.4.3. Parameter Settings

The LR algorithms have varying numbers of parameters: LR has none, LR-L2 and LR-L1 each have one parameter, LR-ElasticNet and LR-GraphNet each have two parameters, while LR-SS1 and LR-SS2 each have four parameters. These parameters are the sparse regularization parameter λ_1, the smooth regularization parameter λ_2, and two parameters δ and ε used for constructing the smooth matrices. In the grid search experiments for parameter optimization, both λ_1 and λ_2 were selected from the range $[10^{-6}, 10^{6}]$, with $\lg(\lambda_1)$ and $\lg(\lambda_2)$ ranging from -6 to 6 with a step size of 0.1.

The parameter δ plays a role in normalizing the distance between features and adjusting the size of the non-zero elements in the adjacency matrix. Since the construction of the smooth matrix mainly focuses on the relationship between the weights of adjacent features, it is appropriate to select δ near 1.

The parameter ε adjusts the sparsity of the adjacency matrix. When $\varepsilon = 1$, the algorithm only considers the correlation between the weights of adjacent features. When $\varepsilon > 1$, the algorithm also considers the correlation between the weights of non-adjacent features, which can extract richer spatial structural information. However, the correlation between feature weights decays rapidly with increasing distance. Therefore, it is appropriate to select ε as 3. Further increasing ε has negligible impact on classification accuracy in practice.

2.4.4. Evaluation Metrics

To comprehensively evaluate the classification performance, we employed five widely used metrics: accuracy, precision, recall, F1 score, and area under the receiver operating characteristic curve (ROC-AUC). Let TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. The metrics are defined as follows. Accuracy measures the overall proportion of correct predictions:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{39}$$

Precision quantifies the proportion of correct positive predictions among all positive predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{40}$$

Recall (also known as sensitivity) measures the proportion of actual positives correctly identified:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{41}$$

The F1 score is the harmonic mean of precision and recall, providing a balanced measure between them:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{42}$$

The ROC-AUC score measures the model's ability to distinguish between classes across different classification thresholds. It is calculated as the area under the curve created by

plotting the true positive rate against the false positive rate at various threshold settings. A perfect classifier has an ROC-AUC of 1, while random guessing yields 0.5. These metrics provide a comprehensive evaluation of binary classification performance.

To quantitatively evaluate the sparsity and smoothness properties of the weight vectors obtained by different methods, we calculated two key metrics, sparsity and smoothness. The first metric is sparsity, which measures how many elements in the weight vector are exactly zero. Let $\mathbf{w} \in \mathbb{R}^d$ denote the weight vector. The sparsity metric is calculated as the percentage of zero elements:

$$\text{Sparsity} = \frac{\|\{\mathbf{w}_i : \mathbf{w}_i = 0\}\|}{d} \times 100\%, \tag{43}$$

where $\|\cdot\|$ denotes the cardinality of a set. Higher sparsity values indicate that more features have been effectively eliminated from the model.

The second metric is smoothness, which quantifies how gradually the weights change across adjacent features. To ensure a fair comparison across methods with different weight value ranges, we first normalize each weight vector by dividing all weights by the maximum absolute weight. The smoothness metric is then defined as the total squared difference between adjacent normalized weights. For vector $\mathbf{w}$, the smoothness metric is calculated as

$$\text{Smoothness} = \frac{1}{\max_j |w_j|^2} \sum_{i=1}^{d-1} (w_i - w_{i+1})^2. \tag{44}$$

Lower smoothness values indicate more gradual transitions between adjacent weights, with perfectly smooth solutions approaching zero.

3. Results

In this section, we present comprehensive experimental results evaluating our proposed methods on both simulated and real-world datasets. Our analysis proceeds in three stages. First, we conduct extensive evaluations of the classification and feature extraction capabilities of various LR algorithms on our primary simulated dataset, using grid search to optimize model parameters. Second, to assess robustness across different noise conditions, we evaluate classification performance on four additional simulated datasets with varying signal-to-noise ratios, employing Bayesian optimization for parameter tuning. Finally, we validate the practical utility of these algorithms by evaluating their classification performance on real-world datasets, again utilizing Bayesian optimization for parameter selection.

3.1. Results on the Simulated Dataset

In the first experiment, we divided the simulated dataset into equal training and testing sets using stratified sampling, ensuring that the proportion of samples from each class remained consistent between the sets. That is, each set contained 1000 samples, with 500 samples from each class. The training set was used to train the LR-SS model and comparative algorithms, while the testing set was reserved for evaluating their performance.

We employed a grid search method to optimize the model parameters for algorithms with one or two parameters. For LR-SS1 and LR-SS2, given the prohibtive computational burden of simultaneously tuning four parameters, we adopted a two-step optimization strategy instead. In the first step, we fixed $\delta = 1$ and $\varepsilon = 3$ based on prior empirical knowledge, while conducting a grid search over λ_1 and λ_2 to identify their optimal values that maximize classification accuracy. In the second step, using these optimal values of λ_1 and λ_2, we performed a focused search over δ and ε to further enhance the model's performance. For each algorithm and each parameter combination, we trained the model

on the training samples and tested it on the test samples. The resulting classification accuracies and weight vectors were recorded to evaluate the classification and feature extraction performance of each algorithm.

3.1.1. Classification Performance on the Simulated Dataset

Figure 4 illustrates the relationship between classification accuracy and the parameters $\lg(\lambda_1)$ and $\lg(\lambda_2)$ across different algorithms. The visualizations reveal distinct patterns in how the accuracy responds to parameter variations, providing insights into each algorithm's sensitivity to its regularization parameters.

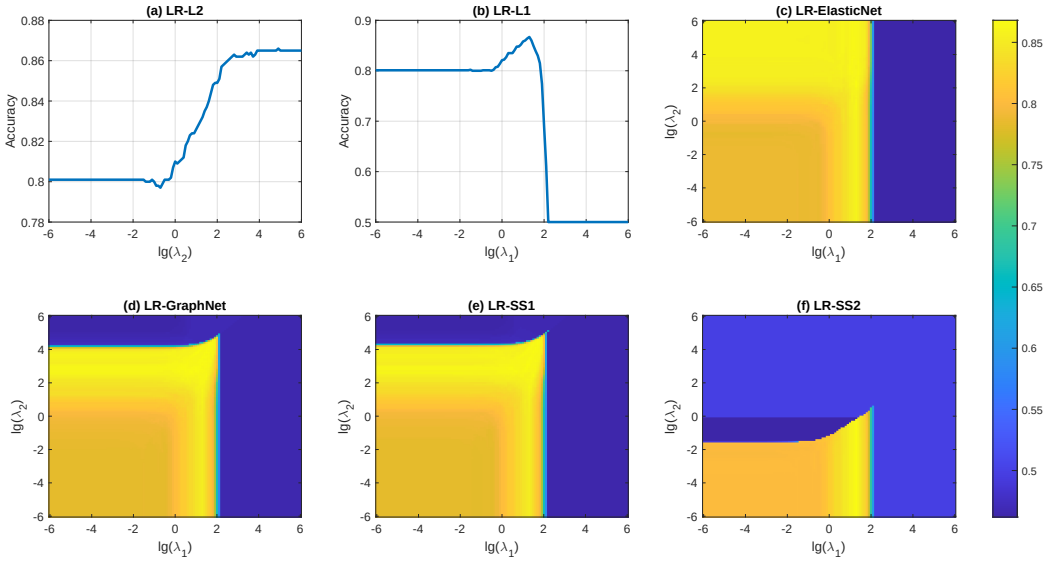

Figure 4. Classification accuracy versus parameters for different algorithms: (**a**) LR-L2, (**b**) LR-L1, (**c**) LR-ElasticNet, (**d**) LR-GraphNet, (**e**) LR-SS1, (**f**) LR-SS2. For LR-SS1 and LR-SS2, the parameters δ and ε are fixed at 1 and 3, respectively.

For LR-L2 (Figure 4a), we observe three distinct regions in the accuracy curve. When $\lg(\lambda_2) \leq -0.3$, the accuracy plateaus at approximately 0.801, indicating a minimal impact of L2 regularization. As $\lg(\lambda_2)$ increases beyond this threshold, the accuracy shows a consistent upward trend, demonstrating the beneficial effect of stronger L2 regularization. Finally, when $\lg(\lambda_2) \geq 3.9$, the accuracy stabilizes around 0.865, suggesting that further increasing the regularization strength yields diminishing returns. The optimal performance is achieved at $\lg(\lambda_2) = 4.9$, with an accuracy of 0.866.

For LR-L1 (Figure 4b), the accuracy exhibits a more complex pattern. When $\lg(\lambda_1) \leq -0.4$, the accuracy remains constant at approximately 0.801, similar to the unregularized case. As $\lg(\lambda_1)$ increases, the accuracy follows an inverted U-shaped curve, first improving as the L1 regularization encourages sparsity, then declining as excessive sparsity begins to degrade performance. The accuracy reaches its peak of 0.867 at $\lg(\lambda_1) = 1.3$, before eventually stabilizing around 0.500 when $\lg(\lambda_1) \geq 2.2$, where the strong L1 regularization forces most coefficients to zero.

For the algorithms with two parameters (Figure 4c–f), we can observe that when $\lg(\lambda_1) \geq 2.2$, the classification accuracy is consistently low, aligning with the results shown in Figure 4b. When $\lg(\lambda_1)$ takes a relatively small value, e.g., $-6 \leq \lg(\lambda_1) \leq 0$, the weight of sparse regularization becomes very low, and LR-ElasticNet approximates LR-L2. As shown in Figure 4c, under these circumstances, the trend of classification accuracy with respect to $\lg(\lambda_2)$ is consistent with the results in Figure 4a.

However, for the other three algorithms, i.e., LR-GraphNet, LR-SS1, and LR-SS2, when $\lg(\lambda_1)$ is very small and sparse regularization has minimal effect, what remains is not L2-

norm regularization but rather smooth regularization with different types. In these cases, the trend of classification accuracy with respect to $\lg(\lambda_2)$ no longer aligns with the results shown in Figure 4a,c. For LR-GraphNet and LR-SS1, when $\lg(\lambda_2) \geq 4.5$, the classification accuracy is generally low in most cases. However, there are exceptions: when $\lg(\lambda_2) \geq 4.5$ and $0 < \lg(\lambda_1) < 2$, some parameter combinations can achieve relatively high classification accuracy. For LR-SS2, when $\lg(\lambda_2) \geq -1.1$, the classification accuracy is generally low in most cases. Similarly, there are exceptions: when $\lg(\lambda_2) \geq -1.1$ and $0 < \lg(\lambda_1) < 2$, some parameter combinations can achieve relatively high classification accuracy. These differences primarily arise from the use of different smooth regularizations.

Among the algorithms with two or more parameters, LR-ElasticNet achieves a classification accuracy of 0.875 at its optimal parameter values of $\lg(\lambda_1) = 1.3$ and $\lg(\lambda_2) = 2.1$. LR-GraphNet shows improved performance, with an accuracy of 0.881, when $\lg(\lambda_1) = 0.7$ and $\lg(\lambda_2) = 2.7$. For the more complex algorithms LR-SS1 and LR-SS2, which each incorporate four tuning parameters (λ_1, λ_2, δ, and ε), we fixed $\delta = 1$ and $\varepsilon = 3$ while optimizing the remaining parameters. Under these conditions, LR-SS1 achieves the highest overall accuracy of 0.882 with $\lg(\lambda_1) = 1.6$ and $\lg(\lambda_2) = 4.0$, while LR-SS2 reaches an accuracy of 0.868 with $\lg(\lambda_1) = 1.3$ and $\lg(\lambda_2) = -0.6$.

Table 5 shows the highest classification accuracies of the seven comparative algorithms and their corresponding optimal parameters.

Table 5. Highest classification accuracy and optimal parameters for different algorithms.

Algorithm	Accuracy	$\lg(\lambda_1)$	$\lg(\lambda_2)$	δ	ε
LR	0.801				
LR-L2	0.866		4.9		
LR-L1	0.867	1.3			
LR-ElasticNet	0.875	1.3	2.1		
LR-GraphNet	0.881	0.7	2.7		
LR-SS1	**0.882**	1.6	4.0	1	3
LR-SS2	0.868	1.3	−0.6	1	3

Values in bold indicate the highest classification accuracy across all methods.

The classification accuracy of LR is the lowest (0.801) among all methods, demonstrating that regularization techniques, whether L2-norm, sparsity, or smoothness constraints, effectively prevent overfitting and enhance the generalization performance of the algorithms. This aligns with statistical learning theory, where regularization helps control model complexity and reduces variance in predictions.

Comparing LR-L2 and LR-L1, which each contain only one regularization term, LR-L1 achieves a slightly higher classification accuracy (0.867) than LR-L2 (0.866). This suggests that the sparsity constraint (L1-norm) is marginally more effective than the L2-norm regularization in this case.

LR-ElasticNet combines L1-norm and L2-norm regularization, achieving a higher classification accuracy (0.875) than both LR-L1 and LR-L2. This improvement demonstrates the benefits of combining different types of regularization. LR-GraphNet further improves upon this by incorporating spatial smoothness constraints, reaching an even higher accuracy of 0.881.

LR-SS1 achieves the highest classification accuracy (0.882) among all methods, showing the effectiveness of combining sparsity with the proposed smooth regularization. However, it is noteworthy that LR-SS2 achieves a lower accuracy (0.868) than LR-SS1, and when LR-SS2 reaches its optimal accuracy, the value of λ_2 is relatively small ($\lg(\lambda_2) = -0.6$). This suggests that for this particular dataset, the specific form of smooth regularization used in LR-SS2 may not provide as much benefit as the form used in LR-SS1.

To investigate the impact of parameters δ and ε on the performance of LR-SS1 and LR-SS2, we fixed $\lg(\lambda_1)$ and $\lg(\lambda_2)$ at their optimal values from Table 5 and varied δ from 0.1 to 10 in increments of 0.1, and ε from 1 to 10 in integer steps. Since we are dealing with one-dimensional signals where the distances between features are integers, ε is restricted to positive integer values. No such restriction applies to δ, allowing it to take decimal values.

These results, shown in Figure 5, indicate reasonable parameter ranges for both algorithms. For LR-SS1, the classification accuracy remains close to the highest accuracy of 0.882 when $\varepsilon = 1$, 2, or 3, or when $\delta \leq 2.7$. For LR-SS2, the classification accuracy stays close to its peak value of 0.868 when $\delta \geq 0.5$ for most cases. These patterns suggest that both algorithms exhibit robustness across certain ranges of parameter values.

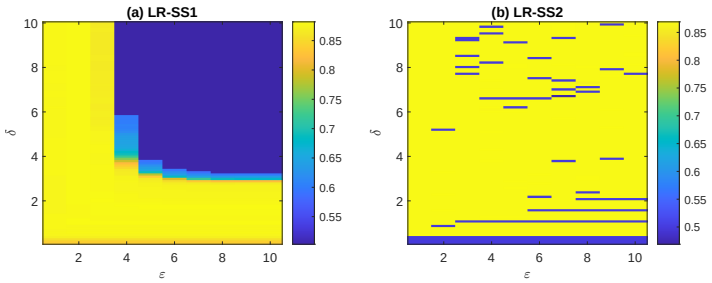

Figure 5. Classification accuracy of (**a**) LR-SS1 and (**b**) LR-SS2 with varying δ and ε, where $\lg(\lambda_1)$ and $\lg(\lambda_2)$ are fixed at their optimal values from Table 5.

It is worth noting that potentially higher classification accuracies could be achieved for LR-SS1 and LR-SS2 through comprehensive optimization of all parameters simultaneously. However, such exhaustive parameter tuning was not conducted in our experiments due to computational constraints. A complete grid search across the four-dimensional parameter space (λ_1, λ_2, δ, and ε) would be prohibitively expensive. Instead, we employed the above two-step optimization approach. While this approach may not guarantee a global optimum, it offers an effective compromise between computational efficiency and model performance, enabling us to systematically analyze the influence of each parameter pair.

3.1.2. Feature Extraction Performance on the Simulated Dataset

Figure 6 presents the weight vectors obtained using optimal parameters from Table 5 for each LR algorithm. The weight vectors from LR and LR-L2 lack both smoothness and sparsity, exhibiting noisy, non-zero values throughout the feature space. In contrast, LR-L1, LR-ElasticNet, LR-GraphNet, LR-SS1, and LR-SS2 demonstrate effective sparsity by reducing numerous weights to zero. Among these sparse solutions, LR-GraphNet and LR-SS1 are particularly noteworthy for their excellent smoothness properties. LR-SS1 proves to be the most effective method, producing weight vectors that closely resemble ideal sinusoidal signals by successfully zeroing out irrelevant regions while maintaining smooth transitions in the sinusoidal regions. This demonstrates an optimal balance between sparsity and smoothness constraints. LR-GraphNet achieves the second-best performance, exhibiting good sparsity and smoothness characteristics, although it retains some non-zero values outside the sinusoidal regions and shows slightly less smooth patterns compared to LR-SS1.

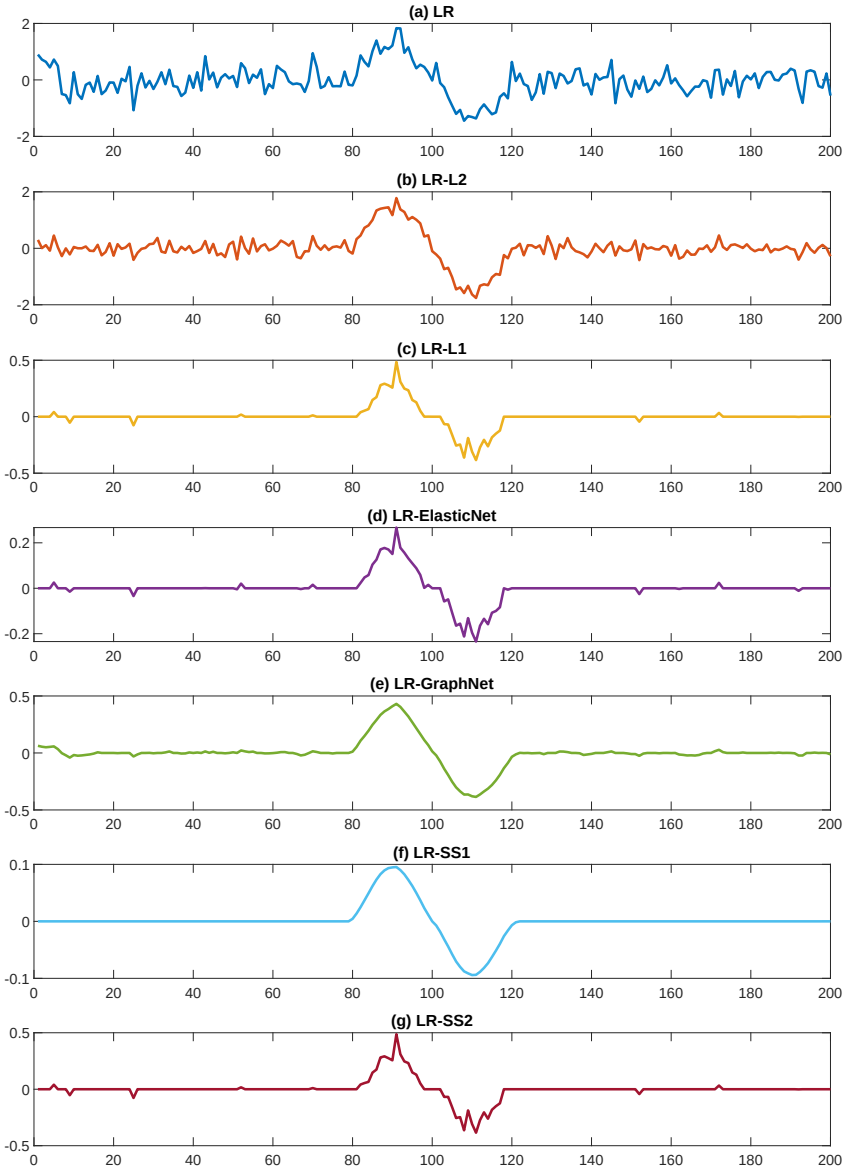

Figure 6. The weight vectors obtained with the optimal parameters from Table 5 for (**a**) LR, (**b**) LR-L2, (**c**) LR-L1, (**d**) LR-ElasticNet, (**e**) LR-GraphNet, (**f**) LR-SS1, and (**g**) LR-SS2.

The remaining algorithms, namely LR-L1, LR-ElasticNet, and LR-SS2, exhibit comparable sparsity characteristics, demonstrating successful identification and preservation of specific patterns while effectively eliminating irrelevant features. The weight pattern obtained by LR-ElasticNet bears a strong resemblance to that of LR-L1, which can be attributed to the dominance of the sparsity regularization over the L2-norm regularization. Similarly, LR-SS2 produces results analogous to LR-L1, primarily due to its small optimal λ_2 value, which substantially reduces the influence of smooth regularization while preserving robust sparsity constraints. However, the patterns extracted through these algorithms lack the refined smoothness characteristics exhibited by LR-SS1 and LR-GraphNet, highlighting the critical role of effective smoothness regularization in accurately capturing the underlying signal structure.

Table 6 presents the sparsity and smoothness metrics for each method. LR and LR-L2 show no sparsity (0%), with all elements being non-zero. Among the sparse methods, LR-SS2 achieves the highest sparsity (80.5%), followed closely by LR-L1 (80.0%), LR-SS1 (79.5%), and LR-ElasticNet (77.0%). LR-GraphNet shows notably lower sparsity (31.5%), indicating it retains more non-zero elements than other sparse methods.

Table 6. Sparsity and smoothness statistics of weight vectors.

Method	Sparsity (%)	Smoothness
LR	0.0	13.1
LR-L2	0.0	4.8
LR-L1	80.0	1.2
LR-ElasticNet	77.0	1.2
LR-GraphNet	31.5	0.5
LR-SS1	79.5	**0.4**
LR-SS2	**80.5**	1.2

Values in bold indicate the best performance: highest sparsity and lowest smoothness values across all methods.

Examining the relationship between sparsity from Table 6 and regularization parameters from Table 5, we observe that sparsity is strongly correlated with the magnitude of λ_1. Generally, larger values of λ_1 lead to increased sparsity in the weight vector. In contrast, the relationship between sparsity and λ_2 is more nuanced. While the impact of λ_2 is considerably less significant compared to that of λ_1, it still influences sparsity to some extent. A notable example is LR-SS1. Despite having the largest λ_1 value among all methods, its relatively large λ_2 value results in a sparsity level that is slightly lower than both LR-SS2 and LR-L1. This suggests that strong smoothness regularization can partially counteract the sparsifying effect of λ_1, leading to solutions that maintain more non-zero elements to achieve smoother transitions in the weight vector.

Regarding smoothness, LR-SS1 demonstrates superior performance, with the lowest smoothness value (0.4), closely followed by LR-GraphNet (0.5). This aligns with the core objectives of these methods, which explicitly incorporate smoothness regularization terms. The enhanced smoothness of LR-SS1 can be attributed to its larger λ_2 parameter compared to LR-GraphNet, resulting in more aggressive smoothness regularization. LR-L1, LR-ElasticNet, and LR-SS2 exhibit moderate smoothness values (all 1.2), with their sparsity-inducing regularization terms effectively zeroing many weights, leading to improved smoothness compared to LR-L2. In the case of LR-SS2, its relatively small λ_2 value limits the impact of the smoothness regularization term, resulting in smoothness characteristics similar to LR-L1 and LR-ElasticNet. The unregularized LR method shows the highest smoothness value (13.1), indicating sharp transitions between adjacent weights and highlighting how any form of regularization tends to improve weight vector smoothness.

The smoothness analysis clearly demonstrates the value of incorporating symmetric smoothness regularization terms. Methods employing explicit smoothness regularizations (especially LR-SS1 and LR-GraphNet) achieve markedly lower smoothness values compared to methods using only sparsity regularization (LR-L1) or no regularization (LR). This indicates that symmetric smoothness regularization terms effectively promote gradual transitions between adjacent weights, contributing to models that are potentially more interpretable and robust. The results suggest that when smooth weight patterns are desired, methods with symmetric smoothness regularization terms should be preferred over those focusing solely on sparsity or using no regularization.

3.2. Classification Performance on Simulated Datasets with Various Signal-to-Noise Ratios

To provide a more rigorous evaluation of the LR algorithms' performance, we conducted additional experiments addressing two key limitations of our previous analysis: the use of a fixed dataset and the separate optimization of parameters for algorithms with four parameters (LR-SS1 and LR-SS2). We employed Bayesian optimization [53,54] for simultaneous parameter tuning and evaluated performance across multiple randomly generated datasets with varying signal-to-noise ratios.

Specifically, we generated four distinct synthetic datasets, each containing two classes with 1000 samples per class. For all datasets, Class 0 samples consisted of pure Gaussian noise (mean 0, variance 1). Class 1 samples were generated by superimposing a sinusoidal signal onto Gaussian noise (mean 0, variance 1). To systematically evaluate algorithm robustness across varying signal-to-noise ratios, we created the four datasets by setting the sinusoidal signal amplitudes to 1, 1/2, 1/4, and 1/8, respectively. The sinusoidal signal was present between time points 80 and 120, consistent with our earlier experiments. For clarity, we refer to these four datasets as Dataset 1 (amplitude = 1), Dataset 2 (amplitude = 1/2), Dataset 3 (amplitude = 1/4), and Dataset 4 (amplitude = 1/8) in order of decreasing signal strength. This systematic variation in signal amplitude allows us to evaluate how each method performs as the classification task becomes increasingly challenging.

For each dataset, we first employed stratified sampling to split the 2000 samples evenly into training and test sets of 1000 samples each. The training set was then further divided using stratified sampling, with 80% used for training and 20% for validation. We used Bayesian optimization [53,54] to tune the parameters, with parameter ranges defined as follows: λ_1 and λ_2 were searched in $[1 \times 10^{-3}, 1 \times 10^{3}]$ on a logarithmic scale, δ in [0.1, 2.0], and ϵ in [1, 5] as integer values. The number of optimization iterations varied based on algorithm complexity: 50 iterations for single-parameter algorithms (LR-L2, LR-L1), 100 iterations for two-parameter algorithms (LR-ElasticNet, LR-GraphNet), and 200 iterations for four-parameter algorithms (LR-SS1 and LR-SS2). After obtaining the optimal parameters, we retrained the models using the combined training and validation sets and evaluated their performance on the test set. To ensure robust statistical results, we repeated this entire process 100 times.

Table 7 presents comprehensive performance metrics across the four synthetic datasets, with signal amplitudes decreasing from 1 to 1/8 (Datasets 1–4). For each method and metric, we report the mean and standard deviation over 100 iterations. The results reveal several important patterns. LR-SS1 and LR-GraphNet consistently outperform other methods across all signal amplitudes, with nearly identical performance metrics. Their advantage is particularly evident as signal strength decreases. Specifically, for Dataset 4 (amplitude = 1/8), LR-SS1 and LR-GraphNet achieve accuracies of (0.605 ± 0.013) and (0.604 ± 0.012), respectively, while all other methods fall below 0.585. The small standard deviations (generally ≤ 0.015) across metrics demonstrate robust performance across different data splits and initializations. As expected, classification performance declines with decreasing signal amplitude, from approximately 0.985 accuracy in Dataset 1 to 0.605 in Dataset 4. However, the relative superiority of LR-SS1 and LR-GraphNet persists across all signal levels.

Figure 7 illustrates these trends in classification accuracy across datasets. The plot reveals excellent performance by all methods on Dataset 1 (accuracy > 0.96), with increasing differentiation as signal strength decreases. Error bars (one standard deviation) demonstrate result consistency across iterations. The visualization emphasizes the sustained performance advantage of LR-SS1 and LR-GraphNet, which becomes more pronounced at lower signal amplitudes. Their narrow error bars further highlight their stability relative to other approaches.

Table 7. Classification performance metrics for different signal amplitudes (mean ± std).

Dataset	Method	Accuracy	Precision	Recall	F1 Score	AUC
1	LR	0.961 ± 0.001	0.966 ± 0.000	0.955 ± 0.002	0.960 ± 0.001	0.993 ± 0.000
	LR-L2	0.983 ± 0.002	0.979 ± 0.002	**0.986 ± 0.003**	0.983 ± 0.002	0.999 ± 0.000
	LR-L1	0.981 ± 0.008	**0.982 ± 0.005**	0.979 ± 0.013	0.981 ± 0.009	0.998 ± 0.002
	LR-ElasticNet	0.984 ± 0.002	0.981 ± 0.002	**0.986 ± 0.003**	0.984 ± 0.002	**0.999 ± 0.000**
	LR-GraphNet	**0.985 ± 0.003**	**0.983 ± 0.002**	**0.986 ± 0.005**	**0.985 ± 0.003**	**0.999 ± 0.001**
	LR-SS1	**0.985 ± 0.003**	**0.983 ± 0.002**	**0.986 ± 0.005**	**0.985 ± 0.003**	**0.999 ± 0.001**
	LR-SS2	0.974 ± 0.025	0.977 ± 0.024	0.972 ± 0.029	0.974 ± 0.026	0.995 ± 0.014
2	LR	0.801 ± 0.000	0.794 ± 0.000	0.812 ± 0.001	0.803 ± 0.000	0.886 ± 0.000
	LR-L2	0.853 ± 0.013	0.842 ± 0.013	0.867 ± 0.013	0.855 ± 0.013	0.928 ± 0.011
	LR-L1	0.853 ± 0.014	0.844 ± 0.015	0.866 ± 0.013	0.855 ± 0.013	0.930 ± 0.010
	LR-ElasticNet	0.861 ± 0.009	0.852 ± 0.011	0.873 ± 0.008	0.863 ± 0.008	0.936 ± 0.007
	LR-GraphNet	**0.871 ± 0.009**	**0.861 ± 0.011**	**0.886 ± 0.009**	**0.873 ± 0.009**	**0.940 ± 0.006**
	LR-SS1	0.870 ± 0.009	0.859 ± 0.010	0.885 ± 0.009	0.872 ± 0.009	**0.940 ± 0.006**
	LR-SS2	0.856 ± 0.010	0.847 ± 0.011	0.868 ± 0.009	0.857 ± 0.009	0.932 ± 0.007
3	LR	0.646 ± 0.000	0.645 ± 0.000	0.650 ± 0.000	0.647 ± 0.000	0.701 ± 0.000
	LR-L2	0.658 ± 0.007	0.652 ± 0.005	0.675 ± 0.015	0.663 ± 0.009	0.718 ± 0.007
	LR-L1	0.659 ± 0.010	0.655 ± 0.009	0.670 ± 0.016	0.663 ± 0.012	0.721 ± 0.014
	LR-ElasticNet	0.663 ± 0.010	0.658 ± 0.010	0.678 ± 0.016	0.668 ± 0.011	0.727 ± 0.014
	LR-GraphNet	**0.690 ± 0.008**	**0.686 ± 0.010**	**0.700 ± 0.008**	**0.693 ± 0.007**	0.767 ± 0.010
	LR-SS1	0.689 ± 0.008	0.684 ± 0.010	**0.700 ± 0.008**	0.692 ± 0.007	**0.768 ± 0.011**
	LR-SS2	0.659 ± 0.011	0.656 ± 0.010	0.670 ± 0.017	0.663 ± 0.013	0.722 ± 0.015
4	LR	0.575 ± 0.000	0.578 ± 0.000	0.554 ± 0.000	0.566 ± 0.000	0.611 ± 0.000
	LR-L2	0.568 ± 0.004	0.569 ± 0.006	0.561 ± 0.011	0.565 ± 0.004	0.614 ± 0.001
	LR-L1	0.583 ± 0.009	0.584 ± 0.009	0.575 ± 0.015	0.579 ± 0.011	0.618 ± 0.007
	LR-ElasticNet	0.580 ± 0.012	0.581 ± 0.012	0.575 ± 0.019	0.578 ± 0.014	0.617 ± 0.009
	LR-GraphNet	0.604 ± 0.012	**0.607 ± 0.012**	0.592 ± 0.018	0.599 ± 0.014	0.646 ± 0.012
	LR-SS1	**0.605 ± 0.013**	**0.607 ± 0.014**	**0.594 ± 0.017**	**0.601 ± 0.015**	**0.649 ± 0.014**
	LR-SS2	0.573 ± 0.028	0.574 ± 0.028	0.566 ± 0.030	0.570 ± 0.028	0.604 ± 0.037

Values in bold denote the highest performance across all methods for each metric within each dataset.

From Table 7 and Figure 7, we can observe that the classification accuracy of LR-SS2 is generally comparable to algorithms without smooth regularizations, including LR-L2, LR-L1, and LR-ElasticNet. These algorithms all show notably lower performance compared to the other two algorithms that incorporate smooth regularizations, namely LR-GraphNet and LR-SS1. Similar conclusions can be drawn from the weight matrices shown in Figure 6. Therefore, the smooth matrix $\mathbf{Q}^{(2)}$ in LR-SS2 is less effective in capturing the temporal or spatial structure of the data for classification and feature extraction purposes, as compared to the smooth matrix $\mathbf{Q}^{(1)}$ in LR-GraphNet and LR-SS1.

When comparing LR-GraphNet and LR-SS1, an interesting observation emerges. While LR-SS1 is theoretically a generalized form of LR-GraphNet and should achieve equal or better classification accuracy with optimal parameter tuning, our experimental results show that their performances are remarkably similar, with LR-GraphNet occasionally achieving marginally better results. This observation warrants discussion from two perspectives. First, it demonstrates the robust performance of the simpler LR-GraphNet model, suggesting that its more constrained parameter space may actually be advantageous in some scenarios. Second, despite employing extensive Bayesian optimization (100 iterations for LR-GraphNet and 200 for LR-SS1), the challenge of simultaneously optimizing four parameters in LR-SS1 versus two in LR-GraphNet highlights the practical limitations of parameter optimization in higher-dimensional spaces, even with sophisticated techniques.

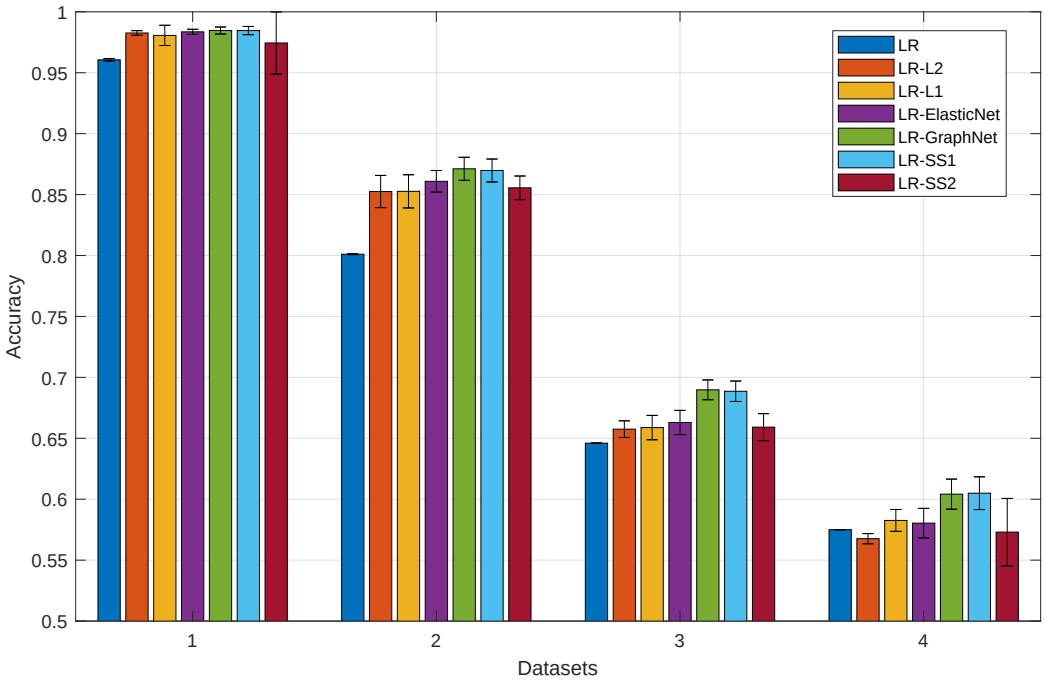

Figure 7. Classification accuracy across datasets with varying signal amplitudes (1, 1/2, 1/4, and 1/8). Results show mean accuracy over 100 iterations, with error bars indicating one standard deviation.

In total, these findings strongly support the effectiveness and robustness of the LR-SS framework, especially LR-GraphNet and LR-SS1, under varying noise conditions. The thorough parameter optimization via Bayesian optimization ensures a fair comparison of each method's capabilities, strengthening our conclusions about their relative performance.

3.3. Classification Performance on Real-World Datasets

Next, we evaluated the classification performance of the LR algorithms on the four real-world datasets. Following our experimental protocol from the synthetic datasets, we split the original training set of each real-world dataset into 80% training and 20% validation data, then utilized Bayesian optimization for parameter tuning. The parameter search ranges and the number of optimization iteration counts remain consistent with those in the synthetic data experiments. Once optimal parameters were determined, we retrained each model on the combined training and validation data with optimal parameters, and then evaluated on the test set. To ensure robust results, we conducted 30 repetitions of this process for the one-dimensional datasets (DistalPhalanxOutlineCorrect and GunPoint). For the computationally intensive image datasets (FashionMNIST and MNIST), which have larger sample sizes and higher dimensionality, we only performed 10 repetitions. Table 8 presents the comprehensive classification performance metrics for all four real-world datasets. Figure 8 illustrates the classification accuracy across the four real-world datasets, with error bars indicating one standard deviation.

The results in Table 8 and Figure 8 reveal several interesting patterns across the four real-world datasets. For the DistalPhalanxOutlineCorrect dataset, LR-GraphNet achieves the best overall performance with the highest accuracy (0.567 ± 0.110), recall (0.701 ± 0.208), F1 score (0.641 ± 0.132), and AUC (0.584 ± 0.098), while LR-SS2 leads in precision (0.606 ± 0.063). The performance advantage of LR-GraphNet and LR-SS1 over traditional methods is particularly notable, suggesting that incorporating structural information is beneficial for bone outline classification.

Table 8. Classification performance metrics for different datasets (mean ± std): (a) DistalPhalanxOutlineCorrect, (b) GunPoint, (c) FashionMNIST, (d) MNIST.

Dataset	Method	Accuracy	Precision	Recall	F1 Score	AUC
(a)	LR	0.516 ± 0.090	0.576 ± 0.074	0.567 ± 0.159	0.568 ± 0.120	0.544 ± 0.061
	LR-L2	0.508 ± 0.085	0.571 ± 0.069	0.556 ± 0.148	0.560 ± 0.111	0.534 ± 0.058
	LR-L1	0.513 ± 0.093	0.573 ± 0.078	0.565 ± 0.166	0.565 ± 0.125	0.539 ± 0.065
	LR-ElasticNet	0.520 ± 0.072	0.582 ± 0.059	0.584 ± 0.130	0.580 ± 0.096	0.541 ± 0.052
	LR-GraphNet	**0.567 ± 0.110**	0.600 ± 0.072	**0.701 ± 0.208**	**0.641 ± 0.132**	**0.584 ± 0.098**
	LR-SS1	0.563 ± 0.108	0.600 ± 0.071	0.684 ± 0.196	0.634 ± 0.129	0.580 ± 0.102
	LR-SS2	0.544 ± 0.078	**0.606 ± 0.063**	0.599 ± 0.119	0.600 ± 0.092	0.558 ± 0.055
(b)	LR	0.757 ± 0.012	0.754 ± 0.012	0.772 ± 0.017	0.763 ± 0.012	0.824 ± 0.008
	LR-L2	0.741 ± 0.058	0.735 ± 0.057	0.767 ± 0.070	0.750 ± 0.057	0.800 ± 0.083
	LR-L1	0.770 ± 0.064	0.752 ± 0.065	0.821 ± 0.067	0.784 ± 0.056	0.821 ± 0.084
	LR-ElasticNet	0.756 ± 0.067	0.746 ± 0.066	0.793 ± 0.097	0.766 ± 0.066	0.808 ± 0.090
	LR-GraphNet	0.773 ± 0.059	0.768 ± 0.064	0.797 ± 0.091	0.779 ± 0.064	0.834 ± 0.059
	LR-SS1	**0.786 ± 0.051**	**0.777 ± 0.058**	**0.819 ± 0.074**	**0.795 ± 0.048**	**0.848 ± 0.072**
	LR-SS2	0.776 ± 0.066	0.761 ± 0.059	0.817 ± 0.077	0.787 ± 0.061	0.835 ± 0.078
(c)	LR	0.973 ± 0.001	0.972 ± 0.003	0.975 ± 0.001	0.973 ± 0.001	0.992 ± 0.001
	LR-L2	**0.978 ± 0.004**	0.980 ± 0.007	0.976 ± 0.002	**0.978 ± 0.004**	**0.995 ± 0.003**
	LR-L1	0.977 ± 0.003	0.979 ± 0.007	0.974 ± 0.002	0.977 ± 0.003	0.993 ± 0.002
	LR-ElasticNet	0.974 ± 0.010	**0.981 ± 0.010**	0.967 ± 0.013	0.974 ± 0.011	0.993 ± 0.006
	LR-GraphNet	0.977 ± 0.005	0.981 ± 0.009	0.974 ± 0.006	0.977 ± 0.005	**0.995 ± 0.003**
	LR-SS1	**0.978 ± 0.004**	0.979 ± 0.007	**0.977 ± 0.002**	**0.978 ± 0.004**	0.994 ± 0.003
	LR-SS2	0.974 ± 0.004	0.979 ± 0.007	0.969 ± 0.008	0.974 ± 0.004	0.994 ± 0.002
(d)	LR	0.991 ± 0.001	0.985 ± 0.002	0.999 ± 0.000	0.992 ± 0.001	0.993 ± 0.001
	LR-L2	**0.997 ± 0.002**	**0.996 ± 0.004**	0.998 ± 0.001	**0.997 ± 0.002**	**0.998 ± 0.003**
	LR-L1	0.994 ± 0.003	0.992 ± 0.006	0.997 ± 0.003	0.994 ± 0.003	0.997 ± 0.003
	LR-ElasticNet	0.996 ± 0.003	0.994 ± 0.005	0.998 ± 0.001	0.996 ± 0.002	**0.998 ± 0.003**
	LR-GraphNet	0.996 ± 0.003	0.995 ± 0.005	**0.999 ± 0.001**	**0.997 ± 0.003**	**0.998 ± 0.003**
	LR-SS1	0.996 ± 0.003	0.995 ± 0.005	**0.999 ± 0.001**	**0.997 ± 0.002**	**0.998 ± 0.003**
	LR-SS2	0.992 ± 0.005	0.991 ± 0.005	0.994 ± 0.008	0.992 ± 0.004	0.996 ± 0.003

Values in bold denote the highest performance across all methods for each metric within each dataset.

For the GunPoint dataset, LR-SS1 demonstrates superior performance across all metrics, achieving the highest accuracy (0.786 ± 0.051), precision (0.777 ± 0.058), recall (0.819 ± 0.074), F1 score (0.795 ± 0.048), and AUC (0.848 ± 0.072). This consistent dominance indicates that LR-SS1's ability to capture temporal dependencies is particularly effective for motion classification tasks.

For the FashionMNIST dataset, both LR-L2 and LR-SS1 share the highest accuracy (0.978 ± 0.004) and F1 score (0.978 ± 0.004), while LR-ElasticNet leads in precision (0.981 ± 0.010) and LR-SS1 in recall (0.977 ± 0.002). LR-L2, LR-GraphNet, and LR-SS1 achieve comparable AUC scores (0.995 ± 0.003, 0.995 ± 0.003, and 0.994 ± 0.003, respectively), suggesting that for this relatively simple binary classification task, i.e., T-shirts vs. trousers, even traditional regularization methods perform well.

For the MNIST dataset, LR-L2 achieves the highest accuracy (0.997 ± 0.002) and precision (0.996 ± 0.004), while LR-GraphNet and LR-SS1 share the highest recall (0.999 ± 0.001). The AUC scores are consistently high (0.998 ± 0.003) across LR-L2, LR-ElasticNet, LR-GraphNet, and LR-SS1, indicating that distinguishing between digits 0 and 1 is a relatively straightforward task where most methods perform exceptionally well.

Overall, these results demonstrate that the LR-SS framework, particularly LR-GraphNet and LR-SS1, performs competitively or superiorly across diverse real-world applications. Its effectiveness is most pronounced in tasks with clear structural dependencies,

such as the temporal patterns in GunPoint and the spatial patterns in DistalPhalanxOutlineCorrect. For simpler classification tasks like binary FashionMNIST and MNIST, the performance differences between methods become less pronounced, though the LR-SS variants still maintain competitive performance.

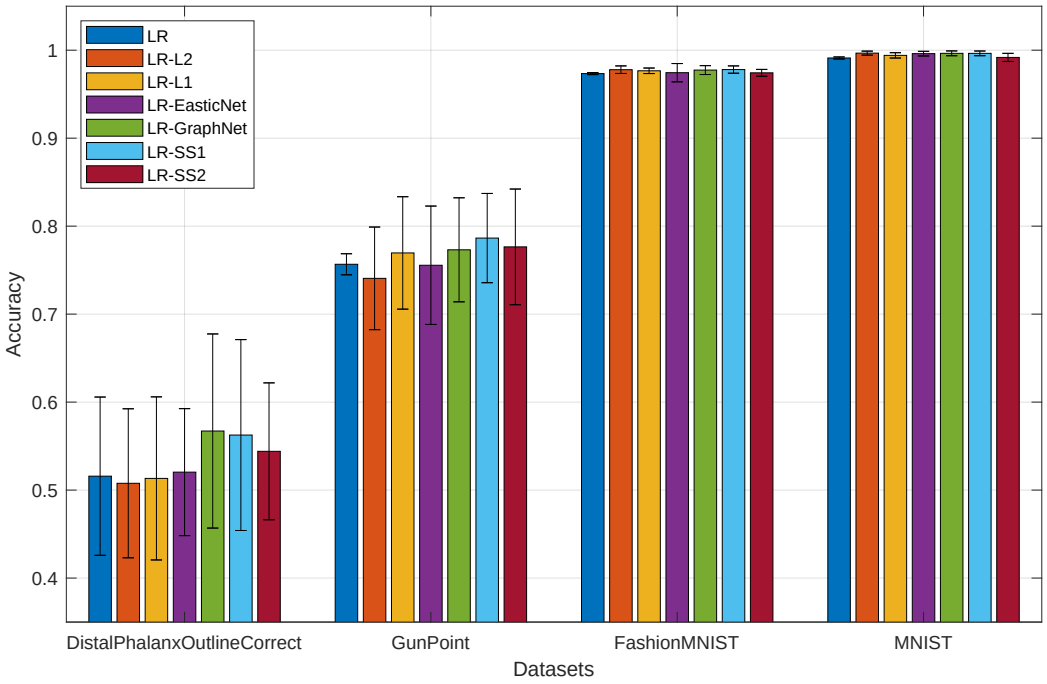

Figure 8. Classification accuracy across four real-world datasets. Results show mean accuracy over 30 iterations for the DistalPhalanxOutlineCorrect and GunPoint datasets and 10 iterations for the FashionMNIST and MNIST datasets, with error bars indicating one standard deviation.

4. Discussion

In this study, we have introduced logistic regression with sparse and smooth regularizations (LR-SS), a novel framework that enhances traditional logistic regression by incorporating both sparsity and smoothness regularizations. The ability to capture temporal and spatial patterns while maintaining sparsity makes LR-SS especially powerful for applications requiring both accurate prediction and interpretable results.

Through parameter adjustment, the proposed framework naturally encompasses several existing algorithms as special cases. Two main variants include LR-SS1, which utilizes the Laplacian-based smooth matrix $\mathbf{Q}^{(1)}$ with extended neighborhood influence, and LR-SS2, which employs the inverse matrix-based smooth matrix $\mathbf{Q}^{(2)}$. LR-GraphNet emerges as a special case of LR-SS1 when $\varepsilon = 1$, representing the scenario where only immediate neighbors are considered in the smoothness constraint.

A significant technical contribution of this study lies in our development of an efficient vectorized iterative solution within the MM framework. The symmetry of our smooth regularization matrices is crucial, ensuring positive semi-definiteness and enabling simplified iterative solutions for both identity matrix and Laplacian matrix-based approaches. This advancement substantially reduces computational overhead while guaranteeing convergence to a local optimum, providing both practical efficiency and theoretical rigor.

4.1. Advantages and Limitations

Our extensive experimental results consistently demonstrate the superior performance of LR-SS variants, particularly LR-SS1 and LR-GraphNet, across both simulated and real-world datasets. The experiments on simulated data were conducted under two different parameter tuning strategies. Using grid search for parameter optimization, LR-SS1 achieved

the highest classification accuracy (0.882), relatively high sparsity (79.5%), and the best smoothness effect (0.4). Most notably, LR-SS1 demonstrated the strongest feature extraction capability by recovering patterns that most closely matched the underlying patterns in the original dataset, followed by LR-GraphNet. When employing Bayesian optimization for parameter tuning, both LR-SS1 and LR-GraphNet consistently outperformed other methods across all signal amplitudes, with nearly identical performance metrics. On real-world datasets, LR-GraphNet achieved the highest accuracy (0.567 ± 0.110) on the challenging DistalPhalanxOutlineCorrect dataset, and LR-SS1 delivered the best performance (0.786 ± 0.051) on the temporal GunPoint dataset. These results validate the framework's effectiveness in balancing sparsity and smoothness constraints while maintaining strong predictive power across diverse application domains.

When comparing LR-GraphNet and LR-SS1, we observe that while both methods achieve comparable classification accuracies, LR-SS1 demonstrates superior feature extraction capabilities in terms of both sparsity and smoothness. Despite its advantages in feature extraction over LR-GraphNet, LR-SS1 has notable limitations that warrant consideration. The computational overhead introduced by the smooth matrix $\mathbf{Q}$ can be substantial, particularly when dealing with large-scale datasets or dense smooth matrices. This increased computational complexity compared to standard sparse logistic regression may impact scalability in resource-constrained environments. Additionally, the framework requires careful tuning of multiple parameters, including the regularization parameters λ_1 and λ_2, as well as the smooth matrix parameters δ and ε. This multi-parameter optimization process is inherently more complex than tuning simpler models, potentially requiring more extensive cross-validation or sophisticated parameter search strategies to achieve optimal performance.

As for LR-SS2 and its corresponding smooth matrix $\mathbf{Q}^{(2)}$, their limitations are evident in both theoretical and empirical analyses. While $\mathbf{Q}^{(2)}$ maintains symmetry, it lacks two crucial properties: the tridiagonal structure that effectively enforces smoothness between adjacent features, and the positive semi-definiteness that guarantees the convexity of the optimization problem. These theoretical shortcomings manifest in practice through consistently inferior classification accuracy and feature extraction quality compared to both LR-GraphNet and LR-SS1, as demonstrated across all experimental datasets. The combination of weaker theoretical properties and poorer empirical performance suggests that LR-SS2 has limited practical utility in real-world applications.

4.2. Implementation Strategy in Practice

Our comparative analysis reveals important practical considerations in choosing between LR-GraphNet and LR-SS1. LR-GraphNet emerges as a robust choice for general applications, offering an excellent balance between model performance and complexity. With only two parameters to tune, it provides competitive classification accuracy while maintaining reasonable computational efficiency. This makes it particularly suitable for scenarios where computational resources are limited.

On the other hand, LR-SS1 represents a more sophisticated approach that can achieve superior performance when computational resources permit. By constructing more complex smoothness models and allowing for finer parameter tuning through its additional parameters (δ and ε), LR-SS1 can match or exceed LR-GraphNet's performance in both classification accuracy and feature extraction quality. The enhanced flexibility in modeling structural relationships comes at the cost of increased computational complexity and more challenging parameter optimization.

This trade-off between model complexity and performance improvement suggests a practical implementation strategy: start with LR-GraphNet as a baseline approach, and if

the application demands higher performance and resources allow, consider upgrading to LR-SS1 for potential gains in both classification and feature extraction capabilities.

4.3. Implications of Model Interpretability

The proposed framework excels in producing interpretable feature vectors through its sparse and smooth characteristics, which is particularly valuable in fields like medicine [2–4] and finance [7] where pattern understanding is crucial. Domain experts can leverage the interpretability of the framework to gain insights that enhance decision-making and practical application.

In the medical field, interpretable models, such as those utilizing sparse and smooth representations, can identify critical features like lesions [2], genes [12,13], brain networks [21], brain regions [31], or other features [16] associated with specific diseases. This not only helps in understanding disease mechanisms but also aids in developing biomarkers for personalized treatment and drug discovery. For instance, by analyzing the factors influencing disease risk, such as age, medical history, or genetic predispositions, physicians can tailor interventions to high-risk patients, thus improving patient outcomes and resource allocation.

In the financial domain, model interpretability facilitates the understanding of complex market patterns and the development of robust investment strategies [7]. Similar features among stocks within the same sector, identified by interpretable coefficients, can reveal sector-wide trends or risks [55]. This clarity enables financial experts to optimize portfolio management and mitigate systemic risks. Furthermore, transparency in applications like credit scoring helps meet regulatory requirements by explaining decisions to stakeholders, thereby fostering trust and reducing legal risks [56].

The impact of interpretability extends beyond improved insights—it drives informed decision-making. By visualizing the relationships between key features and predictions, experts can validate model outputs against domain knowledge, increasing trust in the system. In healthcare, this might involve adjusting treatment plans based on model-driven explanations rather than blind reliance on predictions. In finance, it might involve refining risk management strategies by identifying high-risk customer characteristics. Ultimately, interpretable models not only enhance the effectiveness of domain-specific applications but also promote ethical and transparent use of machine learning or artificial intelligence across industries.

In summary, the LR-SS framework's unique combination of sparsity and smoothness regularization makes it particularly well suited for applications requiring interpretable results. Its ability to produce sparse feature vectors while maintaining smoothness between related features enables clear visualization and understanding of patterns in both medical and financial domains. The framework's interpretability characteristics align well with the growing demand for explainable AI solutions in regulated industries, where transparency and accountability are paramount. Through its balanced approach to feature selection and pattern preservation, LR-SS provides a powerful tool for domain experts to make informed, evidence-based decisions while maintaining compliance with regulatory requirements.

4.4. Future Directions

Future research directions could include the following: 1. Extending the application of LR-SS to additional fields, such as social science [5], finance [7], genomics [12,13], and neuroscience [15,19,21,31], to evaluate its versatility across different types of data. 2. Extending the framework to handle multi-class classification problems [9] or other machine learning paradigms [57]. 3. Developing distributed computing solutions for large-scale applications [10]. 4. Integrating the framework with deep learning architectures [36–39]

to construct interpretable neural networks. 5. Introducing Lp-norm [58,59] into both the sparse and smooth regularization terms to improve the algorithms' classification and feature extraction performance. 6. Investigating advanced metaheuristic optimization techniques for hyperparameter tuning [60,61].

5. Conclusions

In this paper, we have presented LR-SS, a novel framework that advances regularized logistic regression by effectively integrating sparsity and symmetric smoothness constraints. Our framework achieves superior classification performance while maintaining feature interpretability through carefully designed symmetric smoothness regularizations that provide both theoretical guarantees and computational advantages. Through comprehensive experimental evaluation across diverse datasets, we have demonstrated that LR-SS variants, particularly LR-GraphNet and LR-SS1, significantly outperform traditional methods, with especially strong results on data exhibiting temporal or spatial dependencies. The framework's ability to extract meaningful, interpretable features while delivering consistently high predictive accuracy makes it particularly valuable for real-world applications requiring both model performance and explainability.

A key technical contribution is our proposed vectorized iterative solution within the MM framework, which provides both computational efficiency and theoretical soundness. This optimization approach, coupled with the framework's flexibility in handling various types of structural information, establishes LR-SS as a valuable addition to the machine learning toolkit. The methodological principles developed in this work lay a foundation for addressing complex classification challenges across numerous domains.

Looking forward, LR-SS's demonstrated ability to effectively balance sparsity, smoothness, and accuracy while maintaining computational efficiency positions it as a promising approach for future research in machine learning, bioinformatics, neuroscience, and related fields. The framework's success in combining these crucial aspects of modern machine learning suggests its potential for broader impact across the spectrum of data science applications.

Author Contributions: Conceptualization, J.W.; methodology, J.W., P.W., J.S., Y.L. and L.Z.; validation, X.X. and P.W.; writing—original draft preparation, J.W. and P.W.; writing—review and editing, X.X., J.S., Y.L. and L.Z.; visualization, X.X. and L.Z.; funding acquisition, J.W., J.S., Y.L. and L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 31900710, Grant 62403405, and Grant 31600862; in part by the Key Scientific Research Projects of Higher Education Institutions in Henan Province under Grant 25A520020; and in part by the Nanhu Scholars Program for Young Scholars of Xinyang Normal University.

Data Availability Statement: The datasets used in this study are publicly available: (1) the DistalPhalanxOutlineCorrect database can be downloaded from https://www.timeseriesclassification.com/description.php?Dataset=DistalPhalanxOutlineCorrect (accessed on 25 November 2024); (2) the GunPoint database can be downloaded from https://timeseriesclassification.com/description.php?Dataset=GunPoint (accessed on 25 November 2024); (3) the FashionMNIST database can be downloaded from https://github.com/zalandoresearch/fashion-mnist (accessed on 4 December 2024); and (4) the MNIST database can be downloaded from https://yann.lecun.com/exdb/mnist/ (accessed on 4 December 2024). The complete source code used in this study has been made publicly available at https://github.com/yuzhounh/LR-SS (released on 17 January 2025).

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

References

1. Hosmer, D.W., Jr.; Lemeshow, S.; Sturdivant, R.X. *Applied Logistic Regression*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
2. Abdalrada, A.S.; Yahya, O.H.; Alaidi, A.H.M.; Hussein, N.A.; Alrikabi, H.T.; Al-Quraishi, T.A.Q. A Predictive Model for Liver Disease Progression Based on Logistic Regression Algorithm. *Period. Eng. Nat. Sci. (PEN)* **2019**, *7*, 1255–1264.
3. Shipe, M.E.; Deppen, S.A.; Farjah, F.; Grogan, E.L. Developing Prediction Models for Clinical Use Using Logistic Regression: An Overview. *J. Thorac. Dis.* **2019**, *11*, S574. [CrossRef] [PubMed]
4. Cowling, T.E.; Cromwell, D.A.; Bellot, A.; Sharples, L.D.; van der Meulen, J. Logistic Regression and Machine Learning Predicted Patient Mortality from Large Sets of Diagnosis Codes Comparably. *J. Clin. Epidemiol.* **2021**, *133*, 43–52. [CrossRef]
5. Goldstone, J.A.; Bates, R.H.; Epstein, D.L.; Gurr, T.R.; Lustik, M.B.; Marshall, M.G.; Ulfelder, J.; Woodward, M. A Global Model for Forecasting Political Instability. *Am. J. Political Sci.* **2010**, *54*, 190–208. [CrossRef]
6. Bhattacharjee, P.; Dey, V.; Mandal, U. Risk Assessment by Failure Mode and Effects Analysis (FMEA) Using an Interval Number Based Logistic Regression Model. *Saf. Sci.* **2020**, *132*, 104967. [CrossRef]
7. Kemiveš, A.; Ranđelović, M.; Barjaktarović, L.; Đikanović, P.; Čabarkapa, M.; Ranđelović, D. Identifying Key Indicators for Successful Foreign Direct Investment through Asymmetric Optimization Using Machine Learning. *Symmetry* **2024**, *16*, 1346. [CrossRef]
8. Sutton, C.; McCallum, A. An Introduction to Conditional Random Fields. *Found. Trends® Mach. Learn.* **2012**, *4*, 267–373. [CrossRef]
9. Krishnapuram, B.; Carin, L.; Figueiredo, M.A.; Hartemink, A.J. Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 957–968. [CrossRef]
10. Liu, J.; Chen, J.; Ye, J. Large-scale Sparse Logistic Regression. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 547–556.
11. Mohammadi, M.; Atashin, A.A.; Tamburri, D.A. An Efficient Projection Neural Network for ℓ_1-Regularized Logistic Regression. *arXiv* **2021**, arXiv:2105.05449.
12. Shevade, S.K.; Keerthi, S.S. A Simple and Efficient Algorithm for Gene Selection Using Sparse Logistic Regression. *Bioinformatics* **2003**, *19*, 2246–2253. [CrossRef] [PubMed]
13. Serajian, M.; Marini, S.; Alanko, J.N.; Noyes, N.R.; Prosperi, M.; Boucher, C. Scalable De Novo Classification of Antibiotic Resistance of Mycobacterium Tuberculosis. *Bioinformatics* **2024**, *40*, i39–i47. [CrossRef]
14. van Gerven, M.; Hesse, C.; Jensen, O.; Heskes, T. Interpreting Single Trial Data Using Groupwise Regularisation. *NeuroImage* **2009**, *46*, 665–676. [CrossRef] [PubMed]
15. Ryali, S.; Supekar, K.; Abrams, D.A.; Menon, V. Sparse Logistic Regression for Whole-Brain Classification of fMRI Data. *NeuroImage* **2010**, *51*, 752–764. [CrossRef] [PubMed]
16. Zhang, X.; Zhang, Q.; Wang, X.; Ma, S.; Fang, K. Structured Sparse Logistic Regression with Application to Lung Cancer Prediction Using Breath Volatile Biomarkers. *Stat. Med.* **2020**, *39*, 955–967. [CrossRef] [PubMed]
17. Xu, Y.; Du, P.; Robertson, J.; Senger, R. Sparse Logistic Regression on Functional Data. *arXiv* **2021**, arXiv:2106.10583.
18. Klimaszewski, J.; Sklyar, M.; Korzeń, M. Learning ℓ^1-Penalized Logistic Regressions with Smooth Approximation. In Proceedings of the 2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA), Gdynia, Poland, 3–5 July 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 126–130.
19. Grosenick, L.; Klingenberg, B.; Katovich, K.; Knutson, B.; Taylor, J.E. Interpretable Whole-Brain Prediction Analysis with GraphNet. *NeuroImage* **2013**, *72*, 304–321. [CrossRef] [PubMed]
20. de Brecht, M.; Yamagishi, N. Combining Sparseness and Smoothness Improves Classification Accuracy and Interpretability. *NeuroImage* **2012**, *60*, 1550–1561. [CrossRef]
21. Watanabe, T.; Kessler, D.; Scott, C.; Angstadt, M.; Sripada, C. Disease Prediction Based on Functional Connectomes Using a Scalable and Spatially-Informed Support Vector Machine. *Neuroimage* **2014**, *96*, 183–202. [CrossRef]
22. Zhang, C.; Yao, L.; Song, S.; Wen, X.; Zhao, X.; Long, Z. Euler Elastica Regularized Logistic Regression for Whole-Brain Decoding of fMRI Data. *IEEE Trans. Biomed. Eng.* **2017**, *65*, 1639–1653. [CrossRef]
23. Wen, Z.; Yu, T.; Yu, Z.; Li, Y. Grouped Sparse Bayesian Learning for Voxel Selection in Multivoxel Pattern Analysis of fMRI Data. *NeuroImage* **2019**, *184*, 417–430. [CrossRef]
24. Luo, X.; Chang, X.; Ban, X. Regression and Classification Using Extreme Learning Machine Based on L_1-Norm and L_2-Norm. *Neurocomputing* **2016**, *174*, 179–186. [CrossRef]
25. Fuhry, M.; Reichel, L. A New Tikhonov Regularization Method. *Numer. Algorithms* **2012**, *59*, 433–445. [CrossRef]
26. Koh, K.; Kim, S.J.; Boyd, S. An Interior-Point Method for Large-Scale ℓ_1-Regularized Logistic Regression. *J. Mach. Learn. Res.* **2007**, *8*, 1519–1555.
27. Lee, S.I.; Lee, H.; Abbeel, P.; Ng, A.Y. Efficient L_1 Regularized Logistic Regression. In Proceedings of the 21st AAAI Conference on Artificial Intelligence, Boston, MA, USA, 16–20 July 2006; Volume 6, pp. 401–408.
28. Tibshirani, R. Regression Shrinkage and Selection Via the Lasso. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **1996**, *58*, 267–288. [CrossRef]

29. Zou, H.; Hastie, T. Regularization and Variable Selection Via the Elastic Net. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2005**, *67*, 301–320. [CrossRef]
30. Malbasa, V.; Vucetic, S. Spatially Regularized Logistic Regression for Disease Mapping on Large Moving Populations. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011; pp. 1352–1360.
31. Rao, A.; Lee, Y.; Gass, A.; Monsch, A. Classification of Alzheimer's Disease from Structural MRI Using Sparse Logistic Regression with Optional Spatial Regularization. In Proceedings of the 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Boston, MA, USA, 30 August–3 September 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 4499–4502.
32. Meinshausen, N.; Bühlmann, P. High-Dimensional Graphs and Variable Selection with the Lasso. *Ann. Statist.* **2006**, *34*, 1436–1462. [CrossRef]
33. Friedman, J.; Hastie, T.; Tibshirani, R. Sparse Inverse Covariance Estimation with the Graphical Lasso. *Biostatistics* **2008**, *9*, 432–441. [CrossRef] [PubMed]
34. Pothen, A.; Simon, H.D.; Liou, K.P. Partitioning Sparse Matrices with Eigenvectors of Graphs. *SIAM J. Matrix Anal. Appl.* **1990**, *11*, 430–452. [CrossRef]
35. Belkin, M.; Niyogi, P. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Comput.* **2003**, *15*, 1373–1396. [CrossRef]
36. Said, A.; Bayrak, R.; Derr, T.; Shabbir, M.; Moyer, D.; Chang, C.; Koutsoukos, X. NeuroGraph: Benchmarks for Graph Machine Learning in Brain Connectomics. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 6509–6531.
37. Li, X.; Zhou, Y.; Dvornek, N.; Zhang, M.; Gao, S.; Zhuang, J.; Scheinost, D.; Staib, L.H.; Ventola, P.; Duncan, J.S. BrainGNN: Interpretable Brain Graph Neural Network for fMRI Analysis. *Med. Image Anal.* **2021**, *74*, 102233. [CrossRef]
38. Yan, Y.; Zhu, J.; Duda, M.; Solarz, E.; Sripada, C.; Koutra, D. GroupINN: Grouping-based Interpretable Neural Network for Classification of Limited, Noisy Brain Data. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 772–782.
39. Cui, H.; Dai, W.; Zhu, Y.; Kan, X.; Gu, A.A.C.; Lukemire, J.; Zhan, L.; He, L.; Guo, Y.; Yang, C. BrainGB: A Benchmark for Brain Network Analysis with Graph Neural Networks. *IEEE Trans. Med. Imaging* **2022**, *42*, 493–506. [CrossRef] [PubMed]
40. Rudin, L.I.; Osher, S.; Fatemi, E. Nonlinear Total Variation Based Noise Removal Algorithms. *Phys. D Nonlinear Phenom.* **1992**, *60*, 259–268. [CrossRef]
41. Tibshirani, R.; Saunders, M.; Rosset, S.; Zhu, J.; Knight, K. Sparsity and Smoothness via the Fused Lasso. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2005**, *67*, 91–108. [CrossRef]
42. Dohmatob, E.D.; Gramfort, A.; Thirion, B.; Varoquaux, G. Benchmarking Solvers for TV-ℓ_1 Least-Squares and Logistic Regression in Brain Imaging. In Proceedings of the 2014 International Workshop on Pattern Recognition in Neuroimaging, Tubingen, Germany, 4–6 June 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1–4.
43. Hunter, D.R.; Lange, K. A Tutorial on MM Algorithms. *Am. Stat.* **2004**, *58*, 30–37. [CrossRef]
44. Lange, K. *MM Optimization Algorithms*; SIAM: Philadelphia, PA, USA, 2016.
45. 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.
46. Friedman, J.H.; Hastie, T.; Tibshirani, R. Regularization Paths for Generalized Linear Models via Coordinate Descent. *J. Stat. Softw.* **2010**, *33*, 1–22. [CrossRef] [PubMed]
47. Donoho, D.L. De-noising by Soft-Thresholding. *IEEE Trans. Inf. Theory* **1995**, *41*, 613–627. [CrossRef]
48. Cao, F.; Huang, H.; Pietka, E.; Gilsanz, V. Digital Hand Atlas and Web-based Bone Age Assessment: System Design and Implementation. *Comput. Med. Imaging Graph.* **2000**, *24*, 297–307. [CrossRef] [PubMed]
49. Davis, L.M. Predictive Modelling of Bone Ageing. Ph.D. Thesis, University of East Anglia, Norwich, UK, 2013.
50. Ratanamahatana, C.A.; Keogh, E. Three Myths about Dynamic Time Warping Data Mining. In Proceedings of the 2005 SIAM International Conference on Data Mining, Newport Beach, CA, USA, 21–23 April 2005; SIAM: Philadelphia, PA, USA, 2005; pp. 506–510.
51. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
52. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
53. Snoek, J.; Larochelle, H.; Adams, R.P. Practical Bayesian Optimization of Machine Learning Algorithms. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; Volume 25.
54. Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R.P.; De Freitas, N. Taking the Human out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* **2015**, *104*, 148–175. [CrossRef]
55. Kumbure, M.M.; Lohrmann, C.; Luukka, P.; Porras, J. Machine Learning Techniques and Data for Stock Market Forecasting: A Literature Review. *Expert Syst. Appl.* **2022**, *197*, 116659. [CrossRef]

56. Demajo, L.M.; Vella, V.; Dingli, A. An Explanation Framework for Interpretable Credit Scoring. *Int. J. Artif. Intell. Appl. (IJAIA)* **2021**, *12*, 19–38. [CrossRef]
57. Mitchell, T.M.; Mitchell, T.M. *Machine Learning*; McGraw-Hill: New York, NY, USA, 1997; Volume 1.
58. Wang, J. Generalized 2-D principal component analysis by Lp-norm for image analysis. *IEEE Trans. Cybern.* **2015**, *46*, 792–803. [CrossRef]
59. Wang, J.; Xie, X.; Zhang, L.; Chen, X.; Yue, H.; Guo, H. Generalized Representation-based Classification by Lp-norm for Face Recognition. *IAENG Int. J. Comput. Sci.* **2024**, *51*, 104–114.
60. Tang, X.; Liu, S.; Nian, X.; Deng, S.; Liu, Y.; Ye, Q.; Li, Y.; Li, Y.; Yuan, T.; Sun, H. Improved adaptive regularization for simulated annealing inversion of transient electromagnetic. *Sci. Rep.* **2024**, *14*, 5240. [CrossRef] [PubMed]
61. Merikhipour, M.; Khanmohammadidoustani, S.; Abbasi, M. Transportation Mode Detection through Spatial Attention-based Transductive Long Short-Term Memory and Off-Policy Feature Selection. *Expert Syst. Appl.* **2025**, *267*, 126196. [CrossRef]