Learning Nonseparable Sparse Regularizers via Multivariate Activation Functions

Anonymous Authors[1]

Abstract

Sparse regularization is a widely embraced technique in high-dimensional machine learning and signal processing. Existing sparse regularizers, however, are predominantly hand-crafted and often separable, making them less adaptable to data and potentially hindering performance. In this study, we present a novel approach aimed at learning nonseparable (multivariate) sparse regularizers. We leverage the proximal gradient algorithm to transform the challenge of acquiring nonseparable sparse regularizers into the task of learning multivariate activation functions. We further establish the necessary conditions that these activation functions should satisfy. Our contribution culminates in the introduction of MAF-SRL, a deep network designed to learn multivariate activation functions within existing deep learning frameworks. To our knowledge, this research marks the first endeavor to learn nonseparable sparse regularizers. Extensive experiments conducted on benchmark datasets underscore the superiority of regularizers learned through MAF-SRL. They exhibit significantly enhanced performance in terms of both accuracy and sparseness compared to conventional sparse regularizers.

1. Introduction

Sparse regularization is a powerful and widely adopted strategy for tackling challenges in high-dimensional machine learning and signal processing problems. Its effectiveness is well-established through practical applications and rigorous theoretical investigations, as exemplified by the success of techniques like LASSO (Fonti & Belitser, 2017; Kim & Paik, 2019).

One of the remarkable strengths of sparse regularization lies in its dual functionality—it simultaneously performs parameter estimation and feature selection. This unique characteristic produces results that are not only informative but also highly interpretable, as it identifies critical variables. Moreover, it effectively mitigates overfitting by eliminating redundant features. These attributes have propelled sparse regularization to remarkable achievements across diverse domains, spanning machine learning and signal processing. Additionally, extensive theoretical research has bolstered its efficacy, complemented by the development of efficient optimization methods, simplifying its practical implementation.

Despite its widespread adoption, a plethora of sparse regularizers have been introduced to facilitate the generation of sparse solutions. The ℓ_0 (pseudo-)norm, which quantifies the number of non-zero elements, serves as the most intuitive form of sparse regularization, with the primary aim of promoting solution sparsity. Unfortunately, problems involving ℓ_0 norm regularization are typically classified as NP-hard (Natarajan, 1995; Hillar & Lim, 2013; Atserias & Müller, 2020), posing significant computational challenges. Consequently, the ℓ_1 norm has emerged as the predominant surrogate for the ℓ_0 norm [(Candes et al., 2008; Tsagkarakis et al., 2018). This convex alternative substantially simplifies the optimization process, although it is essential to recognize that ℓ_1 regularization, while advantageous, may not consistently yield sufficiently sparse solutions and can introduce notable estimation bias (Fan & Li, 2001; Issa & Gastpar, 2018).

To overcome these limitations, a multitude of alternative sparse regularizers have been proposed and systematically analyzed. These include the smoothly clipped absolute deviation (SCAD) (Fan & Li, 2001; Li et al., 2020), log penalty (Candes et al., 2008; Zhang et al., 2020), capped ℓ_1 (Zhang, 2010; Chen et al., 2019), minimax concave penalty (MCP) (Zhang, 2010; Jiang et al., 2019), ℓ_p penalty with p in the range of $(0, 1)$ (Zhang et al., 2014; Bore et al., 2019), and the difference between ℓ_1 and ℓ_2 norms (Lou et al., 2015; Wu et al., 2018). It is noteworthy that a majority of these regularizers operate in a separable manner, potentially limiting their ability to capture interactions among vector entries and affecting their performance.

In a related context, it is worth mentioning that, to the best

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

of our knowledge, existing sparse regularizers are primarily manually designed. This inherent characteristic raises concerns about their seamless alignment with underlying models to effectively promote sparsity or their suitability for data characteristics to achieve optimal performance. Consequently, practical approaches often involve experimenting with multiple existing sparse regularizers and selecting the most effective one, a process that can be cumbersome in practice.

To address these issues, this paper focuses on learning nonseparable sparse regularizers. Our main contributions can be summarized as follows:

- Leveraging the proximal gradient algorithm, we establish a bridge between nonseparable multivariate regularizers and multivariate activation functions. Notably, a substantial portion of existing sparse regularizers is separable. To our knowledge, this work is the first to tackle the challenge of learning nonseparable (multivariate) sparse regularizers.
- We derive conditions that multivariate activation functions must satisfy to qualify as proper nonseparable sparse regularizers, offering a principled framework for effective regularization.
- Introducing MAF-SRL, a novel deep network that learns multivariate activation functions. This approach allows us to implicitly obtain the desired sparse regularizers, seamlessly integrating them into various machine learning tasks.

Extensive experiments showcase that the sparse regularizers learned by MAF-SRL significantly outperform all existing representative sparse regularizers in terms of both classification accuracy and sparsity.

2. Related Works

The ℓ_1 norm is the most commonly used sparse regularizer (Candes et al., 2008; Tsagkarakis et al., 2018). However, estimation with the ℓ_1 norm is biased (Fan & Li, 2001; Issa & Gastpar, 2018) or the solution may not be sparse enough. This urges researchers to design more general sparse regularizers.

The work in (Fan & Li, 2001; Li et al., 2020) proposed that a good regularizer should result in an estimator with three desired properties: unbiasedness, sparsity and continuity. The smoothly clipped absolute deviation (SCAD) (Fan & Li, 2001; Li et al., 2020) is the first regularizer proven to fulfill these properties (Fan & Li, 2001; Li et al., 2020), whose definition for vector variable $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ is given as $\mathcal{L}(\mathbf{x}; \lambda, \gamma) = \sum_{i=1}^{n} \ell(x_i; \lambda, \gamma)$, in which

$$\ell(x_i; \lambda, \gamma) = \begin{cases} \lambda |x_i|, & \text{if } |x_i| \leq \lambda, \\ \frac{2\gamma\lambda|x_i| - x_i^2 - \lambda^2}{2(\gamma-1)}, & \text{if } \lambda < |x_i| < \gamma\lambda, \\ \lambda^2(\gamma+1)/2, & \text{if } |x_i| \geq \gamma\lambda, \end{cases}$$

where $\lambda > 0$ and $\gamma > 2$. It is obvious that SCAD is a two-parameter function composed of three pieces. Later, (Zhang, 2010; Jiang et al., 2019) proposed a regularizer with two pieces, called minimax concave penalty (MCP). It is formulated as $\mathcal{L}_\gamma(\mathbf{x}; \lambda) = \sum_{i=1}^{n} \ell_\gamma(x_i; \lambda)$, where

$$\ell_\gamma(x_i; \lambda) = \begin{cases} \lambda |x_i| - x_i^2/(2\gamma), & \text{if } |x_i| \leq \gamma\lambda, \\ \gamma\lambda^2/2, & \text{if } |x_i| > \gamma\lambda, \end{cases}$$

for parameter $\gamma > 1$. Log penalty (Candes et al., 2008; Zhang et al., 2020) is a generalization of the elastic net family, which is formulated as $\mathcal{L}(\mathbf{x}; \gamma) = \sum_{i=1}^{n} \ell(x_i, \gamma)$ with

$$\ell(x_i; \gamma) = \frac{\log(\gamma |x_i| + 1)}{\log(\gamma + 1)},$$

where parameter $\gamma > 0$. Through this penalty family, the entire continuum of penalties from ℓ_1 ($\gamma \to 0_+$) to ℓ_0($\gamma \to \infty$) can be obtained (Mazumder et al., 2011; Xu et al., 2017). Capped ℓ_1 is another approximation of ℓ_0 (Zhang, 2008; Chen et al., 2019), whose definition is

$$\mathcal{L}(\mathbf{x}; a) = \sum_{i=1}^{n} \min(|x_i|, a),$$

where $a > 0$. Obviously, when $a \to 0$, $\sum_i \min(|x_i|, a)/a \to \|\mathbf{x}\|_0$. Some other norms in concise forms are also considered as alternatives to improve ℓ_1, such as ℓ_p with $p \in (0, 1)$ (Xu et al., 2012; Sharif et al., 2018), whose expression is

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

The sparse regularizers can also be combined into a new one, for example the ℓ_{1-2} penalty (Yin et al., 2015; Ming et al., 2019), which is the difference between ℓ_1 and ℓ_2 norms, and combined group and exclusive sparsity (CGES) (Yoon & Hwang, 2017). Contour plots of several popular sparse regularizers are displayed in Figure **??**.

Except those involving $\ell_p (p \neq 0, 1)$, all the existing sparse regularizers are separable, i.e., sums of a function of the individual entries of a given vector. Such a separable structure cannot fully exploit the interaction among the entries, thus may not be effective enough for good performance. Moreover, all the above sparse regularizers are hand-crafted, thus cannot adapt to data perfectly. So we aim at learning non-separable sparse regularizers in this paper.

3. The Proposed MAF-SRL Approach

3.1. Connection between Sparse Regularizer and Activation Function

When solving a learning model

$$\min_{\mathbf{x}} \phi(\mathbf{x}), \tag{1}$$

we often need to add a regularizer $g(\mathbf{x})$ to the objective function and solve

$$\min_{\mathbf{x}} \phi(\mathbf{x}) + g(\mathbf{x}) \tag{2}$$

instead, in order to overcome some difficulties in solving (1), e.g., non-unique solutions, or provide prior information of the solution, e.g., sparsity. By adding an appropriate regularizer, the original problem becomes well-posed and solutions with desired properties can be obtained.

When ϕ is L-smooth, i.e.,

$$\|\nabla\phi(\mathbf{x}) - \nabla\phi(\mathbf{y})\|_F \leq L\|\mathbf{x}-\mathbf{y}\|_F,$$

a common algorithm for solving problem (2) is the proximal gradient method (Lu et al., 2015). When applied to (2), the iterations of proximal gradient are as follows:

$$\begin{aligned}
&\mathbf{x}^{(k+1)} \\
&= \arg\min_{\mathbf{x}} \phi\left(\mathbf{x}^{(k)}\right) + \left\langle \nabla\phi\left(\mathbf{x}^{(k)}\right), \mathbf{x}-\mathbf{x}^{(k)} \right\rangle \\
&\quad + \frac{L}{2}\left\|\mathbf{x}-\mathbf{x}^{(k)}\right\|_F^2 + g(\mathbf{x}) \\
&= \arg\min_{\mathbf{x}} \frac{L}{2}\left\|\mathbf{x}-\mathbf{x}^{(k)} + \frac{1}{L}\nabla\phi\left(\mathbf{x}^{(k)}\right)\right\|_F^2 + g(\mathbf{x}).
\end{aligned} \tag{3}$$

We denote $\mathbf{r}^{(k)} = \mathbf{x}^{(k)} - \frac{1}{L}\nabla\phi\left(\mathbf{x}^{(k)}\right)$, then solving (3) requires solving the following optimization problem

$$\mathrm{Prox}_{\alpha g}(\mathbf{r}^{(k)}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{r}^{(k)}\|_F^2 + \alpha g(\mathbf{x}), \tag{4}$$

where $\mathrm{Prox}_{\alpha g}(\cdot)$ is the proximal operator associated with the function $g(\cdot)$, with a parameter $\alpha > 0$. So we can get the solution to (2) by the following iteration:

$$\mathbf{x}^{(k+1)} = \mathrm{Prox}_{L^{-1}g}\left(\mathbf{x}^{(k)} - \frac{1}{L}\nabla\phi\left(\mathbf{x}^{(k)}\right)\right). \tag{5}$$

Note that proximal operators are monotone (Lu et al., 2015) (irrespective of the convexity of g), thus can serve as the activation functions of deep neural networks (DNNs). Conversely, *some* activation functions can be the proximal operators of regularizers, but this inverse correspondence was only exploited in the univariate case (Li et al., 2019; Bibi et al., 2019; Combettes & Pesquet, 2020), possibly due to the fact that up to date only univariate activation functions are in use.

In the case of non-degreasing univariate activation function $\xi(x) : \mathbb{R} \to \mathbb{R}$, we can obtain the univariate regularizer accordingly (Li et al., 2019):[1]

$$g(x) = \int_0^x \left(\xi^{-1}(y) - y\right) dy = \int_0^x \xi^{-1}(y)dy - \frac{1}{2}x^2, \tag{6}$$

such that the proximal operator of g is exactly $\xi(x)$, where $\xi^{-1}(y)$ is the inverse function of $\xi(y)$.

The above relationship between univariate regularizers and univariate activation functions is well known, e.g, (Li et al., 2019; Bibi et al., 2019; Combettes & Pesquet, 2020). Since (6) only gives univariate regularizers, in the following we extend the deduction in the univariate case to the multivariate case.

Note that any multivariate regularizer can be approximated as (Chen & Chen, 1995):

$$\sum_{i=1}^{M} q_i g\left(\mathbf{a}_i^T\mathbf{x} + b_i\right), \tag{7}$$

for appropriate choice of M, g, $\boldsymbol{q}$, $\mathbf{A}$ and $\mathbf{b}$, where $\mathbf{A} = (\mathbf{a}_1, \cdots, \mathbf{a}_M)$, $\boldsymbol{q} = (q_1, \cdots, q_M)^T$ and $\mathbf{b} = (b_1, \cdots, b_M)^T$. This is because (7) is actually a neural network with only one hidden layer.

Since it is unnecessary to model the regularizer very accurately and for ease of computing the parameters in (7), we simply set $M = n$. For making a connection to the proximal operator, inspired by (6), we introduce the parameterization of the regularizer as

$$\mathcal{G}(\mathbf{x}) = \sum_{i=1}^{n} q_i \int_0^{\mathbf{a}_i^T\mathbf{x}+b_i} \widehat{\xi}^{-1}(y)dy - \frac{1}{2}\|\mathbf{x}\|^2, \tag{8}$$

where $\widehat{\xi}(y)$ is a monotonically non-decreasing univariate activation function. Next, we define a multivariate activation function:

$$\xi(\mathbf{x}) = \mathbf{A}^{-T}\left[\widehat{\xi}\left((\mathbf{A}\mathrm{diag}(\boldsymbol{q}))^{-1}\mathbf{x}\right) - \mathbf{b}\right] : \mathbb{R}^n \to \mathbb{R}^n, \tag{9}$$

where $\hat{\xi}$ is applied to the vector $(\mathbf{A}\mathrm{diag}(\boldsymbol{q}))^{-1}\mathbf{x}$ entry-wise. Then we have the following theorem.

Theorem 1 *Given the multivariate activation function ξ in (9). For any $\mathbf{A} = (\mathbf{a}_1, \cdots, \mathbf{a}_n)$, $\boldsymbol{q} = (q_1, \cdots, q_n)^T$ and $\mathbf{b} = (b_1, \cdots, b_n)^T$, such that ξ is well defined, the solution*

[1] A rigorous investigation on the existence of g given ξ is non-trivial, see e.g., Thm. 22.18 of (Bauschke et al., 2011). Since we are not interested in pathological functions, the expression (6) is sufficient for our purpose and has been used in (Li et al., 2019).

to the proximal operator

$$\mathbf{y} = \underset{\mathbf{y}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{y}-\mathbf{x}\|^2 + \mathcal{G}(\mathbf{y}) \tag{10}$$

with $\mathcal{G}$ given in (8) is exactly

$$\mathbf{y} = \xi(\mathbf{x}).$$

Proof: The optimality condition of (10) is:

$$\mathbf{0} \in \sum_{i=1}^{n} q_i \widehat{\xi}^{-1}\left(\mathbf{a}_i^T\mathbf{y} + b_i\right)\mathbf{a}_i - \mathbf{x}. \tag{11}$$

Since $\mathbf{A}$ is invertible, its columns are independent. Further by q_i's all being nonzeros, we may represent $\mathbf{x}$ uniquely as

$$\mathbf{x} = \sum_{i=1}^{n} q_i\beta_i\mathbf{a}_i = \mathbf{A}\,\mathrm{diag}(\boldsymbol{q})\boldsymbol{\beta},$$

where $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_n)^T$. Then $\widehat{\xi}^{-1}\left(\mathbf{a}_i^T\mathbf{y} + b_i\right) = \beta_i, i = 1, \cdots, n$, gives a solution to (11), and we have

$$\mathbf{a}_i^T\mathbf{y} = \widehat{\xi}\left(\beta_i\right) - b_i, \quad i = 1, \cdots, n,$$

which can be written in a matrix form as

$$\mathbf{A}^T\mathbf{y} = \widehat{\xi}(\boldsymbol{\beta}) - \mathbf{b}.$$

Then we get the solution to (10):

$$\begin{aligned}\mathbf{y} =& \mathbf{A}^{-T}(\widehat{\xi}(\boldsymbol{\beta}) - \mathbf{b})\\ =& \mathbf{A}^{-T}\left[\widehat{\xi}\left((\mathbf{A}\,\mathrm{diag}(\boldsymbol{q}))^{-1}\mathbf{x}\right) - \mathbf{b}\right]\\ =& \xi(\mathbf{x}).\end{aligned}$$

■

So we build a connection between the non-separable multivariate regularizer $\mathcal{G}(\mathbf{x})$ and the multivariate activation function $\xi(\mathbf{x})$ via the multivariate proximal operator. For instance, once we choose a multivariate regularizer, the multivariate activation function is determined as its proximal operator. On the other hand, if we choose a multivariate activation function in the form of (9), where the parameters satisfy some conditions (to be specified in Section 3.2 after $\hat{\xi}$ is parameterized), then the multivariate regularizer is determined too. With the above analysis, learning a multivariate regularizer $\mathcal{G}(\mathbf{x})$ is transformed into learning a multivariate activation function $\xi(\mathbf{x})$ that satisfy some conditions.

3.2. Structure of The Activation Function

To learn the activation function $\xi(\mathbf{x})$, we need to learn the parameters: $\mathbf{A}$, $\boldsymbol{q}$, and $\mathbf{b}$, and the univariate function $\hat{\xi}$. For $\mathcal{G}(\mathbf{x})$ to be a sparse regularizer, its proximal operator $\xi(\mathbf{x})$ should be monotone and has to map a neighborhood of $\mathbf{0}$ to $\mathbf{0}$ (Namely, $\mathbf{0} \in \xi^{-1}(\mathbf{0})$).

For the ease of learning, we may first learn

$$\xi(\mathbf{x}) = \widehat{\mathbf{A}}^T[\widehat{\xi}(\mathrm{diag}(\widehat{\boldsymbol{q}})\widehat{\mathbf{A}}\mathbf{x}) - \mathbf{b}]$$

and then obtain $\mathbf{A} = \widehat{\mathbf{A}}^{-1}$ and $\boldsymbol{q} = 1./\widehat{\boldsymbol{q}}$. By defining $\widehat{\mathbf{x}} = \widehat{\mathbf{A}}\mathbf{x}$, we have

$$\begin{aligned}&\langle\xi(\mathbf{x}) - \xi(\mathbf{y}), \mathbf{x} - \mathbf{y}\rangle\\ =&\Big\langle \widehat{\mathbf{A}}^T[\widehat{\xi}(\mathrm{diag}(\widehat{\boldsymbol{q}})\widehat{\mathbf{A}}\mathbf{x}) - \mathbf{b}]\\ &-\widehat{\mathbf{A}}^T[\widehat{\xi}(\mathrm{diag}(\widehat{\boldsymbol{q}})\widehat{\mathbf{A}}\mathbf{y}) - \mathbf{b}], \mathbf{x} - \mathbf{y}\Big\rangle\\ =&\langle\widehat{\xi}(\mathrm{diag}(\widehat{\boldsymbol{q}})\widehat{\mathbf{x}}) - \widehat{\xi}(\mathrm{diag}(\widehat{\boldsymbol{q}})\widehat{\mathbf{y}}), \widehat{\mathbf{x}} - \widehat{\mathbf{y}}\rangle.\end{aligned}$$

Since $\widehat{\xi}$ is non-decreasing and entry-wise, the above is non-negative for all $\mathbf{x}$ and $\mathbf{y}$ iff $\widehat{\boldsymbol{q}} > 0$. So $\xi(\mathbf{x})$ is monotone iff $\widehat{\boldsymbol{q}} > 0$. Note that even with $\widehat{\boldsymbol{q}} > 0$ the regularizer $\mathcal{G}(\mathbf{x})$ may still be non-convex due to $-\frac{1}{2}\|\mathbf{x}\|^2$. For $\mathcal{G}(\mathbf{x})$ to be convex, if $\widehat{\xi}$ is differentiable, we may require $\sum_{i=1}^{n}\frac{q_i}{\widehat{\xi}'\left(\mathbf{a}_i^T\mathbf{y}+b_i\right)}\mathbf{a}_i\mathbf{a}_i^T \succcurlyeq \mathbf{I}$. However, as we do not require $\mathcal{G}(\mathbf{x})$ to be convex, this condition is not enforced during the learning process. Rather, we only require that $\mathcal{G}(\mathbf{x})$ is non-negative. Since $\mathbf{b}$ can compensate the offset of $\widehat{\xi}$, without loss of generality we may fix $\widehat{\xi}(0) = 0$.

It is easy to see that if we choose $\mathbf{b} = \mathbf{0}$, $\widehat{\xi}(x) = 0$ when $x \in [-b, a]$, where $a, b > 0$, then $\xi(\mathbf{x}) = \mathbf{0}$ when $\|\mathbf{x}\|_2 \leq \frac{\min(a,b)}{\max_i\left\{|\widehat{q}_i|\|\widehat{\mathbf{a}}_i\|_2\right\}}$, where $\widehat{\mathbf{A}} = (\widehat{\mathbf{a}}_1, \widehat{\mathbf{a}}_2, \cdots, \widehat{\mathbf{a}}_n)^T$. So it is easy to make $\xi(\mathbf{x})$ all zero. In order to make part of $\xi(\mathbf{x})$ zero, $\mathbf{u} = \widehat{\xi}(\mathrm{diag}(\widehat{\boldsymbol{q}})\widehat{\mathbf{A}}\mathbf{x})$ cannot be all zeros. Rather, most entries of $\mathbf{u}$ should be zeros. In this case, in order that $\xi(\mathbf{x}) = \widehat{\mathbf{A}}^T\mathbf{u}$ is sparse, $\widehat{\mathbf{A}}$ must be a sparse matrix. So in summary, the parameters $\mathbf{A}$, $\boldsymbol{q}$, and $\mathbf{b}$, and $\hat{\xi}$ should satisfy the following conditions:

1. $\widehat{\boldsymbol{q}} > 0$; (12)
2. $\widehat{\mathbf{A}}$ is sparse and invertible; (13)
3. $\widehat{\xi}$ is non-decreasing and $\widehat{\xi}(0) = 0$; (14)
4. $\mathcal{G}(\mathbf{x}) \geq 0$. (15)

In the following, we investigate how to fulfill conditions (14) and (15).

Since $\widehat{\xi}$ is a function, we have to parameterize it first. We use a piecewise linear function to approximate it, denoted as

$\widehat{\xi}_{(\mu_1,\mu_2)}(x)$ with two sets of learnable parameters (μ_1, μ_2):

$$\widehat{\xi}_{(\mu_1,\mu_2)}(x) = \begin{cases} \eta_2 (x-\delta_2) + \eta_1 (\delta_2 - \delta_1), & \delta_2 \le x, \\ \eta_1 (x-\delta_1), & \delta_1 \le x < \delta_2, \\ 0, & -\delta_1 \le x < \delta_1, \\ \eta_1 (x+\delta_1), & -\delta_2 \le x < -\delta_1, \\ \eta_2 (x+\delta_2) + \eta_1 (\delta_1 - \delta_2), & x < -\delta_2, \end{cases}$$

where $x \in \mathbb{R}, 0 \le \delta_1 \le \delta_2$ and $\eta_1, \eta_2 > 0$ are learnable parameters, making $\widehat{\xi}$ non-decreasing. $\mu_1 = (\eta_1, \delta_1)$ and $\mu_2 = (\eta_2, \delta_2)$. With this definition, the inverse function $\widehat{\xi}^{-1}_{(\mu_1,\mu_2)}(y)$ is computed by

$$\widehat{\xi}^{-1}_{(\mu_1,\mu_2)}(y) = \begin{cases} \frac{y-\eta_1(\delta_2-\delta_1)}{\eta_2} + \delta_2, & \eta_1(\delta_2-\delta_1) \le y, \\ \frac{y}{\eta_1} + \delta_1, & 0 \le y < \eta_1(\delta_2-\delta_1), \\ [-\delta, \delta], & y = 0, \\ \frac{y}{\eta_1} - \delta_1, & -\eta_1(\delta_2-\delta_1) \le y < 0, \\ \frac{y-\eta_1(\delta_2-\delta_1)}{\eta_2} - \delta_2 & y < -\eta_1(\delta_2-\delta_1). \end{cases}$$

Therefore, the function $g(x)$ in (6) learned by parameterized activation function $\widehat{\xi}^{-1}_{(\mu_1,\mu_2)}(y)$ can be derived as

$$g(x) = \begin{cases} \left(\frac{1}{2\eta_2} - \frac{1}{2}\right) x^2 \\ + \left(\delta_2 - \frac{\eta_1(\delta_2-\delta_1)}{\eta_2}\right) x \\ + \frac{\eta_1(\eta_1-\eta_2)}{2\eta_2} (\delta_2-\delta_1)^2, & x \ge \eta_1(\delta_2-\delta_1), \\ \left(\frac{1}{2\eta_1} - \frac{1}{2}\right) x^2 + \delta_1 x, & 0 \le x < \eta_1(\delta_2-\delta_1), \\ g(-x). & x < 0. \end{cases} \tag{16}$$

It is observed that $g(x)$ is symmetric about the y-axis. When $x = 0, g(x) = 0$. For $\mathcal{G}(\mathbf{x})$ to be nonnegative, we may require that $g(x) \ge 0$. Due to limited space, we directly give the conditions for (14) and (15) below:

$$\begin{aligned} &\eta_1 > 0, 1 \ge \eta_2 > 0, \\ &\delta_2 \ge \delta_1 \ge \max\left\{0, \tfrac{\eta_1 - 1}{\eta_1}\delta_2\right\}. \end{aligned} \tag{17}$$

The proof of (17) is given in Appendix A. Finally, we obtain that constraints for the learnable parameters are conditions (12), (13) and (17).

3.3. Learning the Activation Function

When an objective function ϕ is given, based on (5) (where g is replace by $\mathcal{G}$) we design a neural network architecture to learn the regularizer $\mathcal{G}$ implicitly. Since by our design the proximal operator $\mathrm{Prox}_{L^{-1}\mathcal{G}}$ is equivalent to a multivariate activation function, we may rewrite (5) as

$$\mathbf{x}^{(k+1)} = \xi_{(\widehat{\mathbf{A}}, \widehat{\boldsymbol{q}}, \mathbf{b}, \mathcal{U})}\left(\mathbf{x}^{(k)} - \frac{1}{L}\nabla\phi\left(\mathbf{x}^{(k)}\right)\right), \tag{18}$$

where $\xi_{(\widehat{\mathbf{A}}, \widehat{\boldsymbol{q}}, \mathbf{b}, \mathcal{U})}$ is the multivariate activation function parameterized by $\widehat{\mathbf{A}}$, $\widehat{\boldsymbol{q}}$, $\mathbf{b}$, and $\mathcal{U}$, in which $\mathcal{U} = \{\mu_1, \mu_2\}$.

Eqn. (18) constitutes the k-th layer of our designed network. The parameters can be learned by the projected gradient method as they are constrained. Since deep learning platforms can compute the gradient by automatic differentiation, we only have to elaborate on how to compute the projection onto the constraints.

Directly projecting parameters $\mathcal{U} = (\eta_1, \eta_2, \delta_1, \delta_2)^T$ onto (17) is difficult. We may first project (η_1, η_2) and then project (δ_1, δ_2) after fixing (η_1, η_2). The projection of (η_1, η_2) is formulated as $\eta_1 = \max\{\eta_1, \epsilon\}$ and $\eta_2 = \min\{\max\{\eta_1, \epsilon\}, 1\}$, where ϵ is a small positive value. After fixing (η_1, η_2), we project (δ_1, δ_2) onto $\mathcal{S}_\delta = \left\{(\delta_1, \delta_2) \mid \delta_2 \ge \delta_1 \ge \max\left\{0, \frac{\eta_1 - 1}{\eta_1}\delta_2\right\}\right\}$.

To be exact, when $0 < \eta_1 \le 1$, the projection $\mathrm{Proj}(\delta_1, \delta_2)$ of (δ_1, δ_2) onto $\mathcal{S}_\delta$ is

$$\mathrm{Proj}(\delta_1, \delta_2) = \begin{cases} (\delta_1, \delta_2), & \delta_1 \ge 0, \delta_2 \ge 0, \delta_1 \le \delta_2, \\ (0, \delta_2), & \delta_1 < 0, \delta_2 > 0, \\ (0, 0), & \delta_2 \le \min\{0, -\delta_1\}, \\ \left(\frac{\delta_1+\delta_2}{2}, \frac{\delta_1+\delta_2}{2}\right), & \delta_1 \ge |\delta_2|. \end{cases} \tag{19}$$

When $\eta_1 > 1$, the projection of (δ_1, δ_2) onto $\mathcal{S}_\delta$ becomes

$$\mathrm{Proj}(\delta_1, \delta_2) = \begin{cases} (\delta_1, \delta_2), & \delta_2 \ge 0, \frac{\eta_1 - 1}{\eta_1}\delta_2 \le \delta_1 \le \delta_2, \\ (\rho_1\delta_1 + \rho_2\delta_2, \rho_2\delta_1 \\ \quad + \rho_3\delta_2), & \frac{\eta_1}{1-\eta_1}\delta_2 < \delta_1 < \frac{\eta_1 - 1}{\eta_1}\delta_2, \\ (0, 0), & \delta_2 \ge 0, \delta_1 \le \frac{\eta_1}{1-\eta_1}\delta_2, \\ (0, 0), & \delta_2 \le \min\{0, -\delta_1\}, \\ \left(\frac{\delta_1+\delta_2}{2}, \frac{\delta_1+\delta_2}{2}\right), & \delta_1 \ge |\delta_2|, \end{cases} \tag{20}$$

where the parameter $\{\rho_1, \rho_2, \rho_3\}$ is given as $\rho_1 = \frac{(\eta_1-1)^2}{\eta_1^2 + (\eta_1-1)^2}$, $\rho_2 = \frac{\eta_1(\eta_1-1)}{\eta_1^2 + (\eta_1-1)^2}$ and $\rho_3 = \frac{\eta_1^2}{\eta_1^2 + (\eta_1-1)^2}$. The proofs of (19) and (20) are given in Appendix B.

When handling condition (12), we actually project $\widehat{\boldsymbol{q}}$ to $\{\mathbf{v} | \mathbf{v} \ge \epsilon\mathbf{1}\}$ to ensure its invertibility, where ϵ is a small positive number and $\mathbf{1}$ is an all-one vector. When handling condition (13), we manually specify the sparsity (percentage of nonzero weights) of $\widehat{\mathbf{A}}$ to be between 30% and 50%, which ensures that $\widehat{\mathbf{A}}$ is both sparse and invertible. Since $\widehat{\mathbf{A}}$ is somewhat random, its invertibility is not an issue when it is not too sparse.

Since our sparse regularizer learning method is through learning a multivariate activation function, we call our method MAF-SRL. Algorithm 1 summarizes MAF-SRL, where the Lipschitz constant of $\nabla\phi$ is also made learnable,

Algorithm 1 Sparse Regularizers Learning via Multivariate Activation Functions (MAF-SRL)

Input: A differentiable function $\phi(\mathbf{x})$, the number N of layers, a set of parameters $\mathbf{x}$ related to training data that need to be solved.
Output: The optimal solution $\mathbf{x}^*$, learned parameters $\widehat{\mathbf{A}}, \widehat{\boldsymbol{q}}, \mathbf{b}, \mathcal{U}$.
Initialize the learnable parameter $\widehat{\mathbf{A}}_{(0)}, \widehat{\boldsymbol{q}}_{(0)}, \mathbf{b}_{(0)}, \mathcal{U}_{(0)}$, Lipschitz constant $L^{(0)}$ and the counter $l = 0$.
Initialize $\mathbf{x}_{(0)} = \xi_{\left(\widehat{\mathbf{A}}, \widehat{\boldsymbol{q}}, \mathbf{b}, \mathcal{U}\right)}\left(\mathbf{x} - \frac{1}{L^{(0)}} \nabla \phi(\mathbf{x})\right)$.
repeat
  for $i = 1$ **to** N **do**
    $\mathbf{x}_{(i)} = \xi_{\left(\widehat{\mathbf{A}}_{(i)}, \widehat{\boldsymbol{q}}_{(i)}, \mathbf{b}_{(i)}, \mathcal{U}_{(i)}\right)}\left(\mathbf{x}_{(i-1)} - \frac{1}{L^{(i-1)}} \nabla \phi\left(\mathbf{x}_{(i-1)}\right)\right)$.
  end for
  Update $\widehat{\mathbf{A}}_{(l)}, \widehat{\boldsymbol{q}}_{(l)}, \mathbf{b}_{(l)}, \mathcal{U}_{(l)}$ with projected gradient descent, where the loss function $\mathcal{L}(\widehat{\mathbf{A}}, \widehat{\boldsymbol{q}}, \mathbf{b}, \mathcal{U}) = \frac{1}{2}\left\|\mathbf{x}_{(N)} - \mathbf{x}\right\|_F^2$.
  Update counter $l = l + 1$.
until convergent
return $\mathbf{x}^* = \mathbf{x}_{(N)}$, $\widehat{\mathbf{A}}, \widehat{\boldsymbol{q}}, \mathbf{b}, \mathcal{U}$.

rather than being manually estimated. After training, we obtain both the parameters of activation function and a sparse output. The sparse regularizer, which has the same parameters as the activation function, can also be obtained but we do not need to write it down as the sparse solution has been obtained.

4. Experiments

In this section, we perform experiments on several real-world public classification datasets. We first choose a backbone network with the ReLU function $f(x) = \max(0, x)$ as the univariate activation function. One-hot encoding is used to encode different classes. We apply the softmax activation function to the output layer and the loss function is the cross entropy. On one specific dataset, we use the same backbone network to keep the comparison fair.

The baselines for comparing with MAF-SRL are the backbone network with the existing hand-craft sparse regularizers added to its loss function. For our MAF-SRL, the ϕ in Algorithm 1 is the loss function of the backbone network.

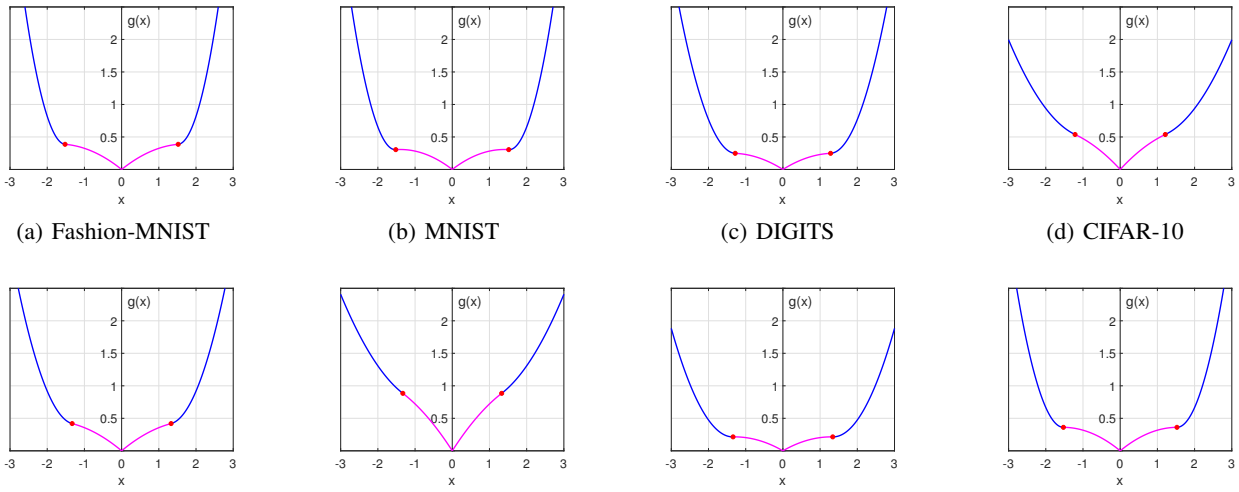
We use Tensorflow to implement the models. We initialize the weights of the models by random initialization according to a normal distribution. The size of minibatch depends on the scale of the datasets. We set the number of layers in MAF-SRL as $N = 16$ and fix the learning rate as $lr = 0.1$. To obtain more reliable results, we run the training process five times in each experiment. Experiments are repeated 20 times, and the averaged performance are reported. We adopt accuracy and weight sparsity (i.e., the ratio of nonzero weights) as the evaluation metrics.

4.1. Baselines and Datasets

We compare our MAF-SRL with several representative state-of-the-art sparse regularizers with ResNet50 as the backbone network: ℓ_1 (Candes et al., 2008; Tsagkarakis et al., 2018), ℓ_{1-2} (Yin et al., 2015; Ming et al., 2019), sparse group lasso (SGL) (Simon et al., 2013), combined group and exclusive sparsity (CGES) (Yoon & Hwang, 2017), smoothly clipped absolute deviation (SCAD) (Fan & Li, 2001), capped-ℓ_1 (Zhang, 2010), log-sum penalty (LSP) (Candes et al., 2008) and minimax concave penalty (MCP) (Zhang et al., 2010).

We select several public classification datasets to conduct experiments:

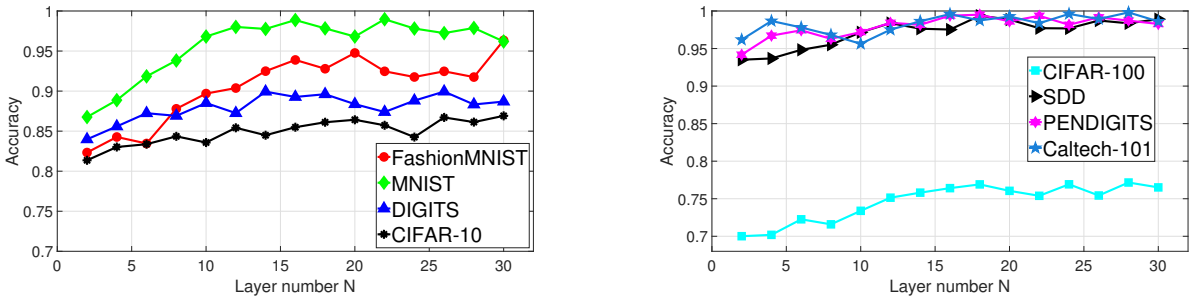
- **Fashion-MNIST**. (Xiao et al., 2017) This dataset consists of a training set with 60,000 instances and a test set with 10,000 examples. Each example is a 28×28 grayscale image, associated with a label from 10 classes.
- **MNIST**. (LeCun et al., 1998) This dataset consists of 70,000 grayscale images of handwritten digits, which can be classified into 10 classes. The numbers of training instances and test samples are 60,000 and 10,000, respectively.
- **DIGITS**. (Netzer et al., 2011) This is a toy dataset of handwritten digits, composed of 1,797 grayscale images.
- **CIFAR-10**. (Krizhevsky et al., 2009) This dataset consists of 60,000 color images in 10 classes, with 6,000 images per class.
- **CIFAR-100**. (Krizhevsky et al., 2009) This dataset comprises 60,000 pixels color images as in CIFAR-10. However, these images can be divided into 100 categories instead of 10 classes and each class has 600 images.
- **Sensorless Drive Diagnosis (SDD)**. (Bayer et al., 2013) This dataset is downloaded from the UCI repository. It contains 58,508 examples obtained under 11 different operating conditions.
- **PENDIGITS**. (Alimoglu & Alpaydin, 1997) This dataset is composed of 10,992 grayscale images of handwritten digits 0-9, where there are 7,494 training instances and 3,498 test samples.
- **Caltech-101**. (Fei-Fei et al., 2004) This dataset consists of images from 101 object categories, and contains from 31 to 800 images per category. Most images are of medium resolution, about 300 × 300 pixels.



(a) Fashion-MNIST (b) MNIST (c) DIGITS (d) CIFAR-10

(e) CIFAR-100 (f) SDD (g) PENDIGITS (h) Caltech-101

Figure 1. The learned univariate function $g(x)$ given in (16) on different datasets for classification tasks. Its associated parameters are as follows: (a) $\eta_1 = 1.37, \eta_2 = 0.22, \delta_1 = 0.46, \delta_2 = 1.57$. (b) $\eta_1 = 1.46, \eta_2 = 0.24, \delta_1 = 0.44, \delta_2 = 1.48$. (c) $\eta_1 = 1.35, \eta_2 = 0.34, \delta_1 = 0.36, \delta_2 = 1.31$. (d) $\eta_1 = 1.41, \eta_2 = 0.62, \delta_1 = 0.62, \delta_2 = 1.49$. (e) $\eta_1 = 1.33, \eta_2 = 0.36, \delta_1 = 0.48, \delta_2 = 1.47$. (f) $\eta_1 = 1.51, \eta_2 = 0.64, \delta_1 = 0.89, \delta_2 = 1.77$. (g) $\eta_1 = 1.34, \eta_2 = 0.45, \delta_1 = 0.33, \delta_2 = 1.33$. (h) $\eta_1 = 1.44, \eta_2 = 0.27, \delta_1 = 0.47, \delta_2 = 1.53$.



(a) FashionMNIST, MNIST, DIGITS, CIFAR-10 (b) CIFAR-100, SDD, PENDIGITS, Caltech-101

Figure 2. The relations among the classification performance (accuracy) and the layer number N of the proposed MAF-SRL.

Table 1. Performance of different methods on the datasets. The weight sparsity is the ratio of nonzero weights in the backbone network.

Dataset	Measure	ℓ_1	ℓ_{1-2}	SGL	CGES	SCAD	capped-ℓ_1	LSP	MCP	MAF-SRL (ours)
Fashion-MNIST	accuracy	0.9124	0.9281	0.8924	0.8873	0.8671	0.8982	0.9031	0.9127	**0.9421**
	weight sparsity	0.2398	0.4363	0.4218	0.2819	0.5728	0.6629	0.2763	0.3397	**0.1537**
MNIST	accuracy	0.9642	0.9538	0.9863	0.9837	0.9824	0.9563	0.9563	0.9623	**0.9921**
	weight sparsity	0.1727	0.2735	0.1029	0.2013	0.1197	0.1126	0.0928	0.3328	**0.0629**
DIGITS	accuracy	0.8638	0.8837	0.8542	0.8837	0.8682	0.8538	0.8772	0.8831	**0.9028**
	weight sparsity	0.3387	0.2928	0.2901	0.4283	0.4419	0.2765	0.5319	0.4019	**0.1774**
CIFAR-10	accuracy	0.8238	0.8188	0.8092	0.8542	0.8452	0.8562	0.8458	0.8229	**0.8759**
	weight sparsity	0.6784	0.5829	0.5429	0.4492	0.5186	0.6294	0.5529	0.3165	**0.2396**
CIFAR-100	accuracy	0.7329	0.7219	0.6872	0.7239	0.6549	0.7129	0.7278	0.7362	**0.7769**
	weight sparsity	0.5587	0.4982	0.8829	0.7623	0.4927	0.6549	0.5498	0.4892	**0.3225**
SDD	accuracy	0.9829	0.9669	0.9539	0.9827	0.9567	0.9632	0.9862	0.9685	**0.9941**
	weight sparsity	0.3092	0.4294	0.2397	0.4962	0.2981	0.3982	0.5729	0.4839	**0.1703**
PENDIGITS	accuracy	0.9852	0.9902	0.9762	0.9683	0.9719	0.9629	0.9739	0.9827	**0.9958**
	weight sparsity	0.6931	0.3397	0.6791	0.3018	0.2973	0.7538	0.5392	0.4492	**0.1778**
Caltech-101	accuracy	0.9733	0.9758	0.9883	0.9901	0.9632	0.9857	0.9683	0.9775	**0.9949**
	weight sparsity	0.3679	0.4133	0.5582	0.6271	0.3036	0.2279	0.3864	0.4272	**0.1762**

4.2. Experimental Results and Analysis

To quantitatively measure the performance of various regularizers, two metrics are utilized, including the prediction accuracy and the sparsity of the weights in the backbone network. A higher accuracy means that the trained network is better for the classification tasks. The smaller ratio of nonzero parameters is, the better sparsity regularization ability is. Table 1 reports the accuracies of all models and their weight sparsities.

As can be seen from Table 1, our model has the best performance when compared with other baselines. On all the datasets, MAF-SRL has both the highest accuracy and the sparsity (lowest percentage of nonzero weights) on all the datasets, showing that our learned multivariate sparse regularizer is indeed effective and can adapt to data better.

We also illustrate the learned univariate function $g(x)$ for different datasets in Figure 1. All learned parameters in $\widehat{\xi}_{(\eta_1,\eta_2,\delta_1,\delta_2)}(x)$ are also reported, where the points $x = \pm\eta_1(\delta_2 - \delta_1)$ are marked in red. We can see that g is not convex, especially on the interval $[-\eta_1(\delta_2 - \delta_1), \eta_1(\delta_2 - \delta_1)]$, and it varies significantly across different datasets, showing that our learnt sparse regularizer can easily adapt to data.

We further investigate the effect of the number N of layers on the performance of our learnt sparse regularizer. The results are shown in Figure 2. The layer number N ranges in $\{2, 4, \cdots, 30\}$, and the learning rate lr is fixed as 0.1. From Figure 2, we can find that the accuracy roughly increases with more layers and becomes stable when the layer number $N > 16$. This is why we set $N = 16$ in previous experiments.

We also apply our framework to the multi-view clustering task for learning an implicit sparse representation. Comprehensive experiments on real-world datasets are presented in Appendix C to validate the superiority of the learned sparse regularizer by MAF-SRL.

5. Conclusion

In this paper, we have proposed MAF-SRL for learning non-separable multivariate sparse regularizer implicitly. We first built a correspondence between multivariate sparse regularizers and multivariate activation functions via the proximal operator. Thus learning a multivariate sparse regularizer is converted to learning a multivariate activation function. We further deduce the conditions that the parameters of multivariate activation function should satisfy. Finally, we apply the project gradient method to accomplish the training of the parameters in the multivariate activation function. Experiments have demonstrated that our proposed MAF-SRL framework achieves higher accuracy and sparser weights than other hand-crafted sparse regularizers.

References

Alimoglu, F. and Alpaydin, E. Combining multiple representations and classifiers for pen-based handwritten digit recognition. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 2, pp. 637–640 vol.2, 1997.

Atserias, A. and Müller, M. Automating resolution is np-hard. *Journal of the ACM (JACM)*, 67(5):1–17, 2020.

Bauschke, H. H., Combettes, P. L., et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

Bayer, C., Enge-Rosenblatt, O., Bator, M., and Mönks, U. Sensorless drive diagnosis using automated feature extraction, significance ranking and reduction. In *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, pp. 1–4, 2013.

Bibi, A., Ghanem, B., Koltun, V., and Ranftl, R. Deep layers as stochastic solvers. In *7th International Conference on Learning Representations, ICLR*, 2019.

Bore, J. C., Ayedh, W. M. A., Li, P., Yao, D., and Xu, P. Sparse autoregressive modeling via the least absolute lp-norm penalized solution. *IEEE Access*, 7:40959–40968, 2019.

Candes, E. J., Wakin, M. B., and Boyd, S. P. Enhancing sparsity by reweighted L1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905, 2008.

Chen, M., Wang, Q., Chen, S., and Li, X. Capped l_1-norm sparse representation method for graph clustering. *IEEE Access*, 7:54464–54471, 2019.

Chen, T. and Chen, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

Combettes, P. L. and Pesquet, J.-C. Deep neural network structures solving variational inequalities. *Set-Valued and Variational Analysis*, pp. 1–28, 2020.

Fan, J. and Li, R. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

Fei-Fei, L., Fergus, R., and Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pp. 178–178. IEEE, 2004.

Fonti, V. and Belitser, E. Feature selection using lasso. *VU Amsterdam Research Paper in Business Analytics*, 30: 1–25, 2017.

Hillar, C. J. and Lim, L.-H. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

Issa, I. and Gastpar, M. Computable bounds on the exploration bias. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 576–580. IEEE, 2018.

Jiang, H., Zheng, W., Luo, L., and Dong, Y. A two-stage minimax concave penalty based method in pruned adaboost ensemble. *Applied Soft Computing*, 83:105674, 2019.

Kim, G.-S. and Paik, M. C. Doubly-robust lasso bandit. *Advances in Neural Information Processing Systems*, 32: 5877–5887, 2019.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

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

Li, J., Fang, C., and Lin, Z. Lifted proximal operator machines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4181–4188, 2019.

Li, Z., Wan, C., Tan, B., Yang, Z., and Xie, S. A fast dc-based dictionary learning algorithm with the scad penalty. *Neurocomputing*, 2020.

Lou, Y., Yin, P., He, Q., and Xin, J. Computing sparse representation in a highly coherent dictionary based on difference of L1 and L2. *Journal of Scientific Computing*, 64(1):178–196, 2015.

Lu, C., Zhu, C., Xu, C., Yan, S., and Lin, Z. Generalized singular value thresholding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Mazumder, R., Friedman, J. H., and Hastie, T. Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138, 2011.

Ming, D., Ding, C., and Nie, F. A probabilistic derivation of lasso and L1-2-norm feature selections. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4586–4593, 2019.

Natarajan, B. K. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Sharif, M., Bauer, L., and Reiter, M. K. On the suitability of lp-norms for creating and preventing adversarial examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

Simon, N., Friedman, J., Hastie, T., and Tibshirani, R. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

Tsagkarakis, N., Markopoulos, P. P., Sklivanitis, G., and Pados, D. A. L1-norm principal-component analysis of complex data. *IEEE Transactions on Signal Processing*, 66(12):3256–3267, 2018.

Wu, S., Li, G., Deng, L., Liu, L., Wu, D., Xie, Y., and Shi, L. $l1$-norm batch normalization for efficient training of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 30(7):2043–2051, 2018.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xu, J., Chi, E., and Lange, K. Generalized linear model regression under distance-to-set penalties. In *Advances in Neural Information Processing Systems*, pp. 1385–1395, 2017.

Xu, Z., Chang, X., Xu, F., and Zhang, H. $l\_\{1/2\}$ regularization: A thresholding representation theory and a fast solver. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1013–1027, 2012.

Yin, P., Lou, Y., He, Q., and Xin, J. Minimization of 1-2 for compressed sensing. *SIAM Journal on Scientific Computing*, 37(1):A536–A563, 2015.

Yoon, J. and Hwang, S. J. Combined group and exclusive sparsity for deep neural networks. In *International Conference on Machine Learning*, pp. 3958–3966, 2017.

Zhang, C.-H. et al. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2): 894–942, 2010.

Zhang, M., Ding, C., Zhang, Y., and Nie, F. Feature selection at the discrete limit. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

Zhang, T. Multi-stage convex relaxation for learning with sparse regularization. *Advances in Neural Information Processing Systems*, 21:1929–1936, 2008.

Zhang, T. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11(3), 2010.

Zhang, Y., Zhang, H., and Tian, Y. Sparse multiple instance learning with non-convex penalty. *Neurocomputing*, 2020.