LassoLayer: Nonlinear Feature Selection by Switching One-to-one Links

Akihito Sudo
Faculty of Informatics
Shizuoka University, Japan
sudo@inf.shizuoka.ac.jp

Teng Teck Hou
ST Engineering Ltd.
Singapore
thteng@stengg.com

Masaki Yamaguchi
yamaguchi.masaki.17
@shizuoka.ac.jp

Yoshinori Tone
JAVIS CO., LTD.
Vietnam.
tone@javis.vn

August 30, 2021

Abstract

Along with the desire to address more complex problems, feature selection methods have gained in importance. Feature selection methods can be classified into wrapper method, filter method, and embedded method. Being a powerful embedded feature selection method, Lasso has attracted the attention of many researchers. However, as a linear approach, the applicability of Lasso has been limited. In this work, we propose LassoLayer that is one-to-one connected and trained by L1 optimization, which work to drop out unnecessary units for prediction. For nonlinear feature selections, we build LassoMLP : the network equipped with LassoLayer as its first layer. Because we can insert LassoLayer in any network structure, it can harness the strength of neural network suitable for tasks where feature selection is needed. We evaluate LassoMLP in feature selection with regression and classification tasks. LassoMLP receives features including considerable numbers of noisy factors that is harmful for overfitting. In the experiments using MNIST dataset, we confirm that LassoMLP outperforms the-state-of-the-art method. The speriority of LassoMLP is significant especially when the number of the training data is small.

1 Introduction

Due to advent of sensor technologies, data can be gathered at ever higher resolution and better precision. Thus, there is the desire to gain invaluable insights from high-dimensional data. A technique called sparse modeling is gaining attention as a method for extracting information from high-dimensional data. Using the sparsity of high dimensional data, sparse modeling can perform calculations in realistic amount of time. Even in situations where the amount of calculation exponentially increases with respect to the number of dimensions, it narrows down to make it easy to extract rules from data.

Lasso [16] is often used to implement sparse modeling. Using Lasso, it is possible to obtain a model expressing the input / output relation after choosing only the required dimensions from the high dimensional data. It can also be regarded as a method of selecting characteristics of data consisting of a set of explanatory variables [5] classified as an embedded feature selection method.

Lasso is a linear regression model in which the loss function has an L1 regularization term and has the property that it is easy to obtain a sparse solution such that the coefficient of the linear regression model becomes zero with the effect of the L1 regularization term. Due to this effect, it becomes possible to construct a model with as few explanatory variables as possible for data of high dimensional inherent in sparseness that many dimensions of explanatory variables do not affect the explained variable. Various Lasso extensions have been studied so far, Group Lasso [12] which collectively tends to have a coefficient of zero in group units, Fused Lasso [17] which can consider spatio-temporal adjacency.

Lasso-based approaches such as standard Lasso, Group Lasso and Fused Lasso are known to not represent nonlinear input-output relationships. Roth et al. [14] suggested to convert variable by nonlinear function to explanatory variable and obtain linear regression model with Lasso for vector obtained by transformation. However, in this method, since features are selected for the converted vector, it is not possible to select the feature of the explanatory variable of the original data. Li et al. [11] proposed a method called Feature Vector Machine (FVM) as a nonlinear method that can perform feature selection on

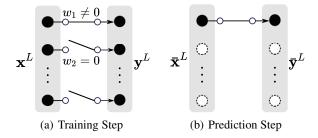(a) Training Step   (b) Prediction Step

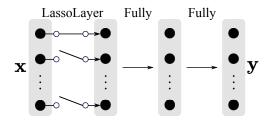Figure 1: The architecture of LassoLayer.

Figure 2: LassoMLP

the original explanatory variable. In the method of Roth et al., non-linear functions are used to convert explanatory variables, whereas in FVM, non-linear functions are used to convert vectors comprising explanatory variables. As a result, features are selected for each dimension of the original explanatory variable. Later, Yamada [18] et al. found that, irregardless of the types of problem, FVM was not able to obtain a good solution when the number of samples falls below the dimension number or when explanatory variable is converted using a nonlinear function. Thus, we propose a method to convert explanatory variables and explanatory variables by using the kernel suggested by [15], because of the limitation of flexibility to capture non-linear relationships among the explanatory variables.

Meanwhile, architectures with deep networks achieve SOTA in various fields [9]. In order to improve the efficiency and accuracy of deep learning, researchers has been proposed various techniques such as convolution layer and pooling layer network structure such as [8], gradient method [7, 19], batch learning method [6] . Because feedforward neural networks of three or more layers can approximate arbitrary nonlinear functions, if this deep learning can be applied to Lasso, it is considered that an embedded type feature extraction method combining non-linearity and deep learning performance can be obtained.

In the present paper, we present LASSOLAYER , which is designed as a filter to achieve feature selection. LASSOLAYER performs feature selection through passing only the features that are effective for prediction to the subsequent layer. Figure 1 depicts the architecture of LASSOLAYER , where the same number of units are connected by one-to-one links. Since the weights are optimized by L1 regularization, the links act as switches that pass only the effective features to the subsequent layers.

By placing LASSOLAYER at the head of the whole network, feature selection for raw inputs is achieved whilst feature selection for the representations in hidden layers can be realized by placing LASSOLAYER in between the hidden layers. Figure 2 shows one of the simplest implementations for feature selection, which we will elaborate on in Section 3.

The presentation of this work continues in Section 2 where a collection of related works are described and contrasted. This is followed by the presentation of our proposed Deep-Lasso method in Section 3. Experiments conducted to evaluate and compare the performance of our proposed Deep-Lasso method are described in Section 4.1. The proposed approach is evaluated using key performance indicators such as accuracy of feature selection and approximation accuracy of the function. Last but not least, Section 5 concludes and summarises this work.

2 Related Work

This section will compare and contrast this work with seminal works addressing the problem of feature selection [3]. In particular, the survey is steered towards the use of deep neural network for feature selection and as a Least Absolute Shrinkage and Selection Operator (Lasso).

Feature selection, also known as variable elimination, is a necessary pre-requisite to many machine learning problems [3]. There are the Filter, Wrapper and Embedded methods. Variable ranking techniques are used as the principle criteria for variable selection in the *filter* methods. A subset of variables is determined using performance of predictor in the *wrapper* methods. The process of eliminating variables is incorporated as part of the training process in the *embedded* methods. Common classifiers used for feature selection tasks are support vector machines (SVM) [1] and radial basis function (RBF) [2]. Variants of the cross validation method method are used for validating the performance of the classifiers calibrated using the selected features.

From a different perspective, feature selection algorithms (FSAs) can be represented using a set of characteristics comprising search organization, generation of successor states and evaluation measures [13]. Feature selection methods such as the Las Vegas Filter (LVF), Las Vegas Incremental (LVI) algorithm, the Relief algorithm, the Sequential Forward Generation (SFG) algorithm, Sequential Backward Generation (SBG) algorithms, Sequential Floating Forward Search algorithm, the Focus algorithm, the Branch & Bound (BB) algorithm and the Quick BB (QBB) algorithm are represented using the suggested approach. It is also suggested in [13] that FSAs be evaluated with respect to particularities such as the relevance, irrelevance, redundance and sample size of the selected features. Four instances from each of the three classical problems are used to generate datasets for evaluating the FSAs. The three classical problems are the *parity* problem, the *disjunction* problem and the *GMonks* problem.

Recently, researchers proposed feature selection methods for nonlinear dataset. HSIC Lasso is a nonlinear Lasso using kernel method [18]. LassoNet selects features by back propagation of a network with residual connection in a network [10]. We will compare our method with these methods in the experimental section.

3 LassoLayer and LassoMLP

In this section, after illustrating how LASSOLAYER works, we introduce LASSOMLP , which equips LASSOLAYER at the first layer. We utilize LASSOMLP to evaluation the performance of LASSOLAYER in the following section.

3.1 Architecture of LassoLayer

The LASSOLAYER consists of two layered units, those are linked one-by-one as illustrated in Fig. 1. A bias unit is not necessary. Let n be the number of the LASSOLAYER's input units. There are the same number of the output-side unites. We denote by $\mathbf{w}$ be the weights in the LassoLayer. The weight $\mathbf{w}$ is a n dimensional vector.

Suppose that the LASSOLAYER receives the inputs $\mathbf{x}^L$. The input-side unites activates the inputs by σ_{in}, and pass them to the output-side units through the weighted one-to-one link; The output units receive the vector $\mathbf{w} \odot \sigma_{\mathrm{in}}(\mathbf{x}^L)$, where we denote Hadamard product by $\odot$. Finally, the output units activate them, and thus, the LASSOLAYER produces the n dimensional output $\mathbf{y}^L$:

$$\mathbf{y}^L = \sigma_{\mathrm{out}} \left(\mathbf{w} \odot \sigma_{\mathrm{in}}(\mathbf{x}^L) \right). \tag{1}$$

3.2 Training of LassoLayer

For simplicity, we assume that there is exactly one LASSOLAYER in the network to illustrate how the network is trained although it is straightforward to extend the training scheme to the network equipped with more than two LASSOLAYER . Let θ be the trained parameter except for $\mathbf{w}$. We denote by $E_{\theta,\mathbf{w}}$ the loss function for the network when the network does not contain LASSOLAYER .

To realize the feature switching, we add a L_1 penalty of $\mathbf{w}$ to a loss function:

$$L_{\theta,\mathbf{w}} = E_{\theta,\mathbf{w}} + \lambda\|\mathbf{w}\|_1 \tag{2}$$

where λ is a hyperparameter that determines the size of the L1 penalty. Since the derivative of the penalty term vanishes, only the first term affects the gradient to update the parameters θ. Meanwhile, the weights $\mathbf{w}$ in LASSOLAYER is trained with the L1 penalty, which shrinks less important inputs to zero. The zero weights filters the corresponding weights. Thus, the one-to-one links in LASSOLAYER are suppose to switch the inputs by the penalty term. Traditional regularization techniques often employ the L_1 norm or L_2 norm of all parameters as a regularization term, but L_1 norm of only

LASSOLAYER's weights is required to achieve feature selection by LASSOLAYER

For $E_{\theta,\mathbf{w}}$, one can select any loss function known effectively working for the whole network such as mean square error, cross entropy. Additional regularization term is also permissible.

We introduce a heuristic for stable training; kicking out the zero-valued weights to become non-zero during in the early phase of training. To prevent false negative on feature selection, the zero-valued weights in LASSOLAYER are kicked out to inspect whether the corresponding features are really ineffective for a prediction. Precisely, during the first m^{kick} epoch, the zero-valued weights in LASSOLAYER are updated to have some non-zero values by the probability of ρ. The updated values are chosen from either $-\delta$ or δ by the probability 0.5. Since these strategies are found by empirical observations and increase the effort of hyperparameter tuning, finding the theoretically optimal strategies is a important direction for future work.

Further, it is known that some heuristics is required for making parameters exactly zero by L1 penalties. Without such heuristics, even if loss functions have the L_1 term , the trained parameter is unlikely to become zero. Hence, in practice, we employ a method such as Duchi's method [4] to realize zero-valued parameters. In Duchi's method, when the absolute value of a parameter is smaller than the hyperparameter λ, the parameter value is forced to become zero.

3.3 Prediction with LassoLayer

There are multiple ways to make predictions with a network equipped with LassoLayer. In this section, we introduce two methods. One is to make predictions on the raw network that has been trained, and the other is to make predictions using only the features corresponding to the top-k weights of the LassoLayer. For the former method, the predictions of the network are given through computing the output of LassoLayer with the equation (1). Let w^k and w_i be the k-th largest element of $\mathbf{w}$ and i-th element of $\mathbf{w}$, respectively. For the latter method, the output of LassoLayer is determined as follows:

$$\mathbf{y}^L = \sigma_{\mathrm{out}} \left(\vec{\mathbb{1}}_{\geq w^k} \odot \sigma_{\mathrm{in}}(\mathbf{x}^L) \right), \tag{3}$$

where $\vec{\mathbb{1}}_{\geq w^k}$ is the n-dimensional vector of which j-th element is determined by

$$\mathbb{1}^i_{\geq w^k} = \begin{cases} 1 & \text{if } w_i \geq w^k \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

3.4 LassoMLP

To evaluation LASSOLAYER , we implement the network as the sequence of a LASSOLAYER and a three-layered MLP, as illustrated in Fig. 2. We call it LASSOMLP and believe this architecture is the simplest one for
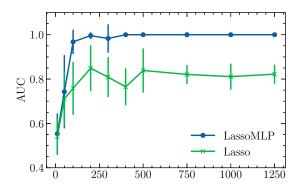
Figure 3: Feature selection performance with the synthetic dataset.

LASSOLAYER to achieve the nonlinear feature selection of the input data. The LASSOLAYER receives and filters raw inputs to pass only effective features to MLP. The MLP performs a nonlinear transformation on the filtered data. passed to the LASSOLAYER , and filtered to pass through the filtered data are processed.

Suppose that the LASSOMLP receives the inputs $\mathbf{x} \in \mathbb{R}^{n^{in}}$ and produces the outputs $\mathbf{y} \in \mathbb{R}^{n^{out}}$. Both the activation functions σ_{in} and σ_{out} are the identity map. Thus, by Eq. (1), the LASSOLAYER passes the filtered signal $\mathbf{y}^L = \mathbf{w} \odot \mathbf{x}$. We denote by n^h the number of the hidden units of the MLP. Let the weights of first and second layer in the MLP be W_1 and W_2. The weights W_1 and W_2 are $n^{in} \times n^h$ and $n^h \times n^{out}$ matrices. The outputs $\mathbf{y}$ are determined as follows:

$$\mathbf{y} = \sigma_{out} \left[W_2 \, \sigma_h \left(W_1 \mathbf{y}^L \right) \right] \qquad (5)$$

4 Experiments

4.1 Experiments with Nonlinear Synthetic Data

4.1.1 Setting

To confirm that the proposed method can perform feature selection and function approximation on non linear data, we compare LassoMLP with Lasso using synthetic dataset. The number of dimensions P of input data is 256 dimensions, and the dimension of output is 1. That is, the data D is a set of the pairs $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathbb{R}^{256}$ and $y \in \mathbb{R}$. Each element of $\mathbf{x}$ is generated from the standard normal distribution of which the mean is 0 and the variance is 1. To generate the value of y corresponding to $\mathbf{x}$, we utilized the following nonlinear functions:

$$y = \sin(x_1) \exp(-x_2) + (x_3 - 0.2)^2, \qquad (6)$$

where (x_1, x_2, x_3) is the first three elements of $\mathbf{x}$. We denote the number of the pairs in D by N.

The number of of hidden units was 100. The hyperparameter of LassoMLP is as follows: $\lambda = 0.0001$, $\rho = 0.1$, and $\delta = 0.1$. For Lasso, $\lambda = 0.1$. The optimizer was

the stochastic gradient descent (SGD) with the learning rate 0.1. In addition, we use the mini batch learning with the batch size determined by $\min(80, D)$. The total number of training epoch is 6000. Of those epochs, the kicking strategy was applied in the first 1000 epochs; namely $m^{kick} = 1000$. The activation function of the hidden units is ReLU.

To evaluate the feature-selection performance, we measured AUC score.

4.1.2 Result

Fig. 3 shows ACU of the feature selection task for varying number of the training data. While Lasso and LassoMLP are competitive when the number of the training data is small, LassoMLP outperforms Lasso when the training data is greater than 100. Then, although AUC of Lasso is approximately 0.8, that of LassoMLP is approximately 1.0. This result suggests that LassoMLP performs better than the vanilla Lasso for the nonlinear dataset if sufficient dataset is provided.

4.2 Feature Selection Performance with Real-world Dataset

We compare the performance of LassoMLP with baseline methods in terms of feature selection.

4.2.1 Setting

The task is the feature selection from MNIST data so as to classify the digits accurately with fewer features. The dimension of the images is 784 (28 x 28). We evaluated LassoMLP and baseline methods with the classification accuracy with the selected features. Firstly, the features are selected by each method using only training data. Secondly, Random Forest (RF) are trained by the training data including only selected features. Lastly, the accuracy is evaluated by the test data that also include only selected features. We adopt this evaluation scheme by following Ref. [10].

We added the additional non-relevant features to MNIST images. The number of dimension is 5000. The non-relevant features are sampled from the standard Gaussian. The dataset was divided randomly into the training data and the test data with varying split size from 0.05% to 5% training samples. Then, while the number of dimension is 5784, the number of the training samples is from 35 to 3500. The variation of the numbers of the trained samples are shown in the title of Fig. 4 (i.e. $35, 70, 140, \cdots, 3500$). We employed this unbalance split because we are interested in the feature selection in sparse dataset. The validation data is not necessary because we used the default set of the hyper parameters of SVM and RF was employed across all the baselines.

The hyper parameters of LassoMLP are as follows: batch size is 80, $\lambda = 0.005$, $\rho = 0.1$, $\delta = 0.25$. The optimizer is SGD with the learning rate 0.1.
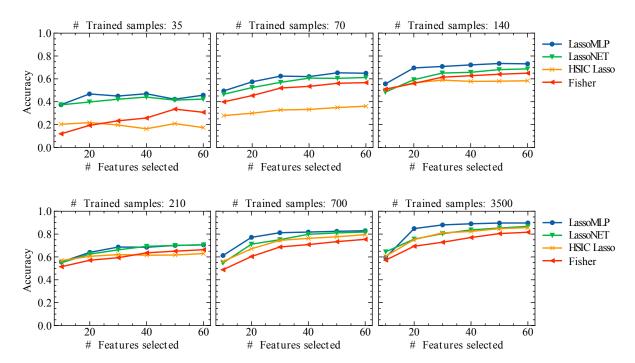
Figure 4: Feature selection performance with MNIST dataset.

The baseline methods include LassoNet, HSIC Lasso, PFA, and Fisher score. We follows Ref. [10] to choose the baselines. For LassoNet and HSIC Lasso, we made use of the implementation in their original github repositories. We used the default hyper paramters of LassoNet and HSIC Lasso in their implementations. We implemented PFA using the implementations of PCA and k-means in scikit-learn. We used the implementation of Fisher score in scikit-feature.

4.2.2 Results

Fig. 4 shows the classification accuracy by RF using only selected features with varying training samples. The selected features varies from 10 to 60.

LassoMLP outperformed all baseline methods for all the number of features and trained samples at least in this experimental setting with two exceptions, the pairs of (# trained sample and # features) are (210, 40) and (3500, 10).

4.3 Generalization Performance by Feature Selection

4.3.1 Setting

To show that feature selection can improve generalization performance, we compared the accuracy of LassoMLP with that of discriminators using all features in a setting with very little training data. The task was binary classification, classifying 1 from 4 or 1 from 7 in MNIST. We used as input a 5784-dimensional vector created by combining a 784-dimensional vector of digit images with a

5000-dimensional vector sampled from a standard Gaussian distribution. The baseline methods are MLP, SVC, and RF. The hyperparameters for LassoMLP are as follows: $\lambda = 0.005$, $\rho = 0.1$, $\delta = 0.25$. The batch size and the number of hidden units is 200 and 100 for both LassoMLP and MLP. The optimizer is SGD with the learning rate 0.1 for them.

4.3.2 Result

As shown in Table 1, LassoMLP performed competitively with SVC and RF when the number of training data was 75, and outperformed SVC and RF when the number of training data was 15 or less, even though the discriminator was MLP.

We believe that the reason why the accuracy of LassoMLP is relatively high even with a small number of training data is that the number of features is sufficiently smaller than the number of training data due to feature selection. Indeed, as shown in Table 2, the average number (ratio) of features used by LassoMLP ranges from 3.5 (0.4%) to 16.1 (2.1%). Furthermore, almost all irrelevant features of dimension 5000 have been removed.

5 Conclusion

We proposed a neural architecture for embedded feature selection method, which is an important element technology of sparse modeling which attracts attention in recent years by extracting information from high dimensional data. The difference between the proposed method and the conventional neural network is that the first layer com-

Table 1: Accuracy on MNIST784+Gauss5000 for the average (variance) of running experiments 10 times.

(a) Classify 1 from 4

the number of training data	4 (0.03%)	7 (0.05%)	15 (0.1%)	75 (0.5%)
Multilayer Perceptron	0.512 (±0.021)	0.481 (±0.0)	0.558 (±0.166)	0.729 (±0.222)
Random Forest	0.501 (±0.02)	0.526 (±0.041)	0.704 (±0.152)	**0.975** (±0.009)
Support Vector Classifier	0.627 (±0.146)	0.573 (±0.119)	0.8 (±0.178)	0.967 (±0.009)
LassoMLP (ours)	**0.744** (±0.083)	**0.692** (±0.079)	**0.941** (±0.006)	0.955 (±0.002)

(b) Classify 1 from 7

the number of training data	4 (0.03%)	7 (0.05%)	15 (0.1%)	75 (0.5%)
Multilayer Perceptron	0.507 (±0.045)	0.517 (±0.077)	0.481 (±0.0)	0.835 (±0.218)
Random Forest	0.514 (±0.014)	0.51 (±0.049)	0.814 (±0.176)	**0.974** (±0.005)
Support Vector Classifier	0.557 (±0.068)	0.61 (±0.191)	0.737 (±0.177)	0.963 (±0.013)
LassoMLP (ours)	**0.754** (±0.122)	**0.771** (±0.118)	**0.941** (±0.007)	0.954 (±0.002)

Table 2: Ratio (number and standard deviation) of remained features on MNIST784+Gauss5000 for running experiments 10 times.

(a) Classify 1 from 4

#training data	4 (0.03%)	7 (0.05%)	15 (0.1%)	75 (0.5%)
MNIST Features	0.021 (16.1±8.77)	0.009 (7.0±1.63)	0.004 (3.5±2.42)	0.014 (10.8±16.7)
Gaussian Features	0.001 (4.3±3.33)	0.0 (0.0±0.0)	0.0 (0.0±0.0)	0.001 (5.6±15.41)

(b) Classify 1 from 7

#training data	4 (0.03%)	7 (0.05%)	15 (0.1%)	75 (0.5%)
MNIST Features	0.016 (12.2±9.09)	0.007 (5.3±2.54)	0.006 (5.0±5.44)	0.006 (5.0±5.93)
Gaussian Features	0.001 (5.4±5.34)	0.0 (0.0±0.0)	0.0 (0.0±0.0)	0.0 (0.0±0.0)

bines one-to-one and that the loss function includes the L_1 regularization term of the first layer weight, It is possible to naturally apply research results of deep learning which is actively studied.

We conducted several experiments to evaluate LassoMLP. In the first experiment, we confirmed that LassoMLP works better for nonlinear dataset than vanilla Lasso. In the second experiment, we showed that LassoMLP outperforms LassoNet in feature selection task, which means our method achieves state-of-the-art in our setting. Lastly, we illustrated that the feature selection by LassoMLP improve the generalization performance through showing that LassoMLP predicts better than the full-feature classifier.

In the future, we will discuss theoretical analysis of this method, comparison of performance with nonlinear methods, benchmark in real data such as gene microarray, application to neural networks with various structures such as convolutional neural network, stochastic gradient We plan to evaluate the performance in detail and expand the application range by using gradient method other than law etc.

References

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[2] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[3] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers and Electrical Engineering*, 40:16–28, 2014.

[4] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

[5] I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh. *Feature extraction: foundations and applications*. Series Studies in Fuzziness and Soft Computing. Physica-Verlag, Springer, 2006.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing inter-

nal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[9] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521.7553:436–444, 2015.

[10] I. Lemhadri, F. Ruan, L. Abraham, and R. Tibshirani. Lassonet: A neural network with feature sparsity. *Journal of Machine Learning Research*, 22(127):1–29, 2021.

[11] F. Li, Y. Yang, and E. P. Xing. From LASSO regression to feature vector machine. In *Advances in Neural Information Processing Systems*, pages 779–786, 2006.

[12] L. Meier, S. V. D. Geer, and P. Bühlmann. The group LASSO for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[13] L. C. Molina, L. Belanche, and À. Nebot. Feature selection algorithms: A survey and experimental evaluation. In *Proceedings of the IEEE International Conference on Data Mining*, pages 306–313. IEEE, 2002.

[14] V. Roth. The generalized LASSO. *IEEE Transaction on Neural Networks*, 15(1):16–28, 2004.

[15] I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *The Journal of Machine Learning Research*, 2:67–93, 2002.

[16] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[17] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused LASSO. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67:91–108, 2005.

[18] M. Yamada, W. Jitkrittum, L. Sigal, E. P. Xing, and M. Sugiyama. High-dimensional feature selection by feature-wise kernelized LASSO. *Neural computation*, 26(1):185–207, 2014.

[19] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.