A neural network with feature sparsity

Ismael Lemhadri, Feng Ruan* and Robert Tibshirani
Department of Statistics, and Department of Biomedical Data Science
Stanford University

July 30, 2019

Abstract

We propose a neural network model, with a separate linear (residual) term, that explicitly bounds the input layer weights for a feature by the linear weight for that feature. The model can be seen as a modification of so-called residual neural networks to produce a path of models that are feature-sparse, that is, use only a subset of the features. This is analogous to the solution path from the usual Lasso (ℓ_1-regularized) linear regression. We call the proposed procedure `LassoNet` and develop a projected proximal gradient algorithm for its optimization. This approach can sometimes give as low or lower test error than a standard neural network, and its feature selection provides more interpretable solutions. We illustrate the method using both simulated and real data examples, and show that it is often able to achieve competitive performance with a much smaller number of input features.

1 Introduction

In many applications, neural networks achieve state-of-the-art accuracy. This technology has provided near-human performance on many prediction tasks, and left deep marks on entire fields of business and science, to the extent that large computational and engineering efforts are routinely dedicated to neural network training and optimization. However, neural networks are sometimes criticized for their complexity and lack of interpretability. There are many arguments that favor simple models over more complex ones. In many applications (including medical diagnosis, insurance and banking products, flight control, among others) interpretation of the underlying model is a critical requirement. On the other hand, traditional statistical tools, including simple linear models, remain popular because they are simple and explainable, with cheap, efficient computational tools are readily available. As a consequence, there has been growing interest in simplifying and understanding neural networks all the while preserving their performance. This paper attempts to bridge the gap between interpretability and performance, by enforcing sparsity on the input features fed into the neural network.

The Lasso estimate (Tibshirani (1996)) is perhaps the most prominent tool for enforcing sparsity in linear models, thus easing their interpretability. The sparsity pattern provides a direct way to identify the most significant features of the model. Just like the Lasso, our proposed model is controlled by a penalty factor, that is chosen through cross-validation or from a validation set. In addition to improving explainability, a feature sparse structure may improve the model's generalization ability, as was confirmed by our experiments.

Figure 1 shows an example, using the Boston housing data consisting of 506 observations of house prices and 13 predictors. We divided the data into training and test sets of size 404 and 102 respectively, and then added 13 Gaussian noise predictors. The test error for the Lasso, a single hidden layer neural

*The first two authors contributed equally.

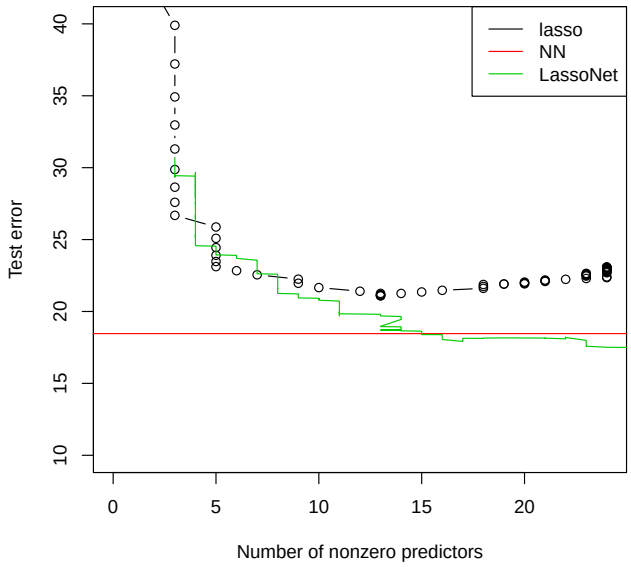

Figure 1. *Boston housing data with 13 noise predictors added. The test error shown for a (single hidden layer) neural net is the minimum achieved over 200 epochs.*

network and our proposed procedure `LassoNet` [1] are shown. We see that the new procedure achieves the minimum test error at around 13 predictors, and does as well as the standard neural network. Furthermore, it turned out that 11 of the first 13 predictors chosen came from the list of real rather than the artificial predictors.

The outline of this paper is as follows. In Section 2 we briefly review related work on this problem. Section 3 introduces the proposed `LassoNet` procedure. In Sections 4 and 5 we examine performance of the procedure for simulated and real datasets. The pathwise optimization procedure is detailed in 6, and we end with some extensions and discussion in Section 7.

2 Short review of past work

Sparse neural architectures are a topic of active research. And while much of the literature has focused on full network sparsity (for e.g., computational reasons, or memory and time efficiency), this brief review focusses on feature-sparsity. Considering many of today's high-dimensional datasets, e.g. from biomedical or computer vision problems (among many others), it is of particular interest to find a small subset of input features that contribute most of the explanatory power Guyon & Elisseeff (2003).

Pruning procedures. The simple and most popular approach relies on so-called pruning methods, that share the common trait of being performed *after* model training. Verikas & Bacauskiene (2002) proposed to examine the output sensitivity to input changes due to the removal of individual features. However, sensitivity-based approaches suffer from treating features individually. It would be combinatorially intractable to consider larger groups of features, yet feature importance unavoidably depends on what other features are present in the model, cf section 3.3 of Guyon & Elisseeff (2003). In Han et al. (2015), a thresholding approach is proposed. First, the network is trained in its entirety

[1] We recently discovered a procedure proposed by Webber et al. (2018) with an R language implementation of the same name. However it treats a completely different problem, a linear model with network constraints. With apologies to these authors, the name "LassoNet" seems very appropriate for the method proposed here.

with an ℓ_2 penalty. Then, weights whose magnitude are below a certain threshold are set to zero, and the model is trained from scratch, again, on the remaining features. In addition to lengthening the training time, such a method requires a separate pruning process. However, unifying training and selection would be a desirable property of the model, especially from the selective inference perspective. Reed (1993) provides a more comprehensive survey of pruning-related methods.

Weight regularization. A somewhat more principled approach is to enforce regularization through penalization. The ℓ_1 norm acts as a convex proxy for the ℓ_0 norm, directly penalizing the number of active features, and originating in the LassoLasso estimator. There has been growing interest in methods that produce structured sparsity Wen et al. (2016), Yoon & Hwang (2017), Scardapane et al. (2017). These methods make use of the group lasso penalty, a variant of the lasso which induces entire groups of neural weights to be set to zero. Taking the example of Scardapane et al. (2017), which enforces sparsity by applying a group lasso penalty on a specific grouping of the weights, the authors apply a group penalty to the set of weights that originate from a given feature. However, this formulation being non-convex at the origin, standard gradient-based methods are insufficient, and a thresholding step after optimization is generally required to obtain precise sparsity.

3 The LassoNet proposal

3.1 Formulation

We start with the usual regression data (x_{ij}, y_i), $i = 1, 2, \dots n; j = 1, \dots p$, with y_i being a quantitative outcome. Let $x_i = (x_{i1} \dots x_{ip})$. We assume the model

$$y_i = \beta_0 + \sum_j x_{ij}\beta_j + \sum_{k=1}^{K} [\alpha_k + \gamma_k \cdot f(\theta_k^T x_i)] + \epsilon_i \tag{1}$$

with $\epsilon \sim (0, \sigma^2)$. Here f is a monotone, nonlinear function such as a sigmoid or rectified linear unit, and each $\theta_k = (\theta_{1k}, \dots \theta_{pk})$ is a p-vector. Our objective is to minimize

$$J(\beta, \Theta) = \frac{1}{2} \sum_i (y_i - \hat{y}_i)^2 + \lambda \sum_j |\beta_j| + \gamma \sum_{jk} |\theta_{jk}|; \text{ subject to } |\theta_{jk}| \le M \cdot |\beta_j| \ \ \forall j, k. \tag{2}$$

Extensions to other problems such as classification are straightforward, by replacing the squared error term with a binomial or multinomial log-likelihood.

Note that there are three tuning parameters: the main parameter λ controls the complexity and sparsity of the fitted model. The secondary parameter M controls the relative complexity between the linear and nonlinear parts of the model, with $M = 1$ being the default. Typically we will vary λ over a dense grid that spans the spectrum from dense and sparse, and try a few values for M, e.g $M \in [0.2, 0.5, 1, 2, 5]$. We use a validation set or cross-validation to choose optimal values for λ and M. Finally, the term $\gamma \sum_{jk} |\theta_{jk}|$, while not needed to control complexity, can provide additional sparsity for the θ_{jk}. While our algorithm can handle values $\gamma \ge 0$, we set $\gamma = 0$ in all examples of the paper.

The objective function (2) is non-convex, and we outline our procedure for its optimization for each fixed pair of parameters (λ, M) in Section 6.

3.2 Strategy for path optimization

Our strategy is to optimize (2) for each fixed value M, over a wide path of λ values. This kind of strategy is used in Lasso (ℓ_1)-regularized linear regression, where coordinate descent is often used for each fixed value of λ (see Friedman et al. (2010)). In this approach, optimization is carried out on a fine grid from sparse (large λ) to dense (small λ), using warm starts in $\hat{\beta}$ for each λ. This is very effective, since the sparse models are easier to optimize and the sparse regime is often the main (or

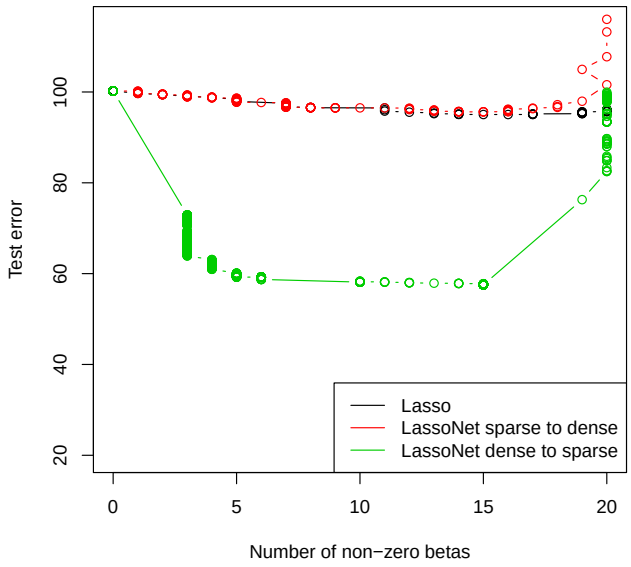

Figure 2. *Simulated example from model (3), comparing the Lasso, and the two optimization strategies for* `LassoNet`*. The dense to sparse strategy gives superior performance.*

only) region of interest. Further, the convexity of the Lasso problem ensures that we can find the global minimum for each λ.

Somewhat to our surprise, we have found that in the (non-convex) `LassoNet` problem, this sparse-to-dense approach does not work well, and can get caught in poor local minima. Instead, we find that a dense-to-sparse strategy is far more effective.

Figure 2 shows an example. With $n = 100, p = 20$ we generated standard independent Gaussian features, and the outcome from the model

$$y_i = \sum_{j=1}^{10} x_{ij}\beta_j + g(x_i) + \sigma \cdot \epsilon_i \tag{3}$$

with $g(x) = 4x_1^2 + 2x_4 - 2x_2^2 + 3x_3^2$, $\epsilon_i \sim N(0, \sigma^2)$ and $\sigma = 3$, giving a signal-to-noise ratio of about 3. We generated a test set with 1000 observations for assessing the model performance, set $M = 3$ and the number of hidden units $K = 50$. Figure 2 shows the test error for the Lasso, and `LassoNet` using both strategies. We see that only the dense-to-sparse approach captures the nonlinear signal. Figure 3 shows the path of solutions as λ is varied.

4 Simulation study

We carried out a simulation study to compare the performance of `LassoNet` to the Lasso and a standard single hidden layer neural network. For further exploration, we also implemented a two stage sparse neural network procedure with a linear term, defined as follows:

1. Compute the Lasso path of solutions for the linear model.
2. For each of k equally spaced solutions along the path (we used $k = 10$)

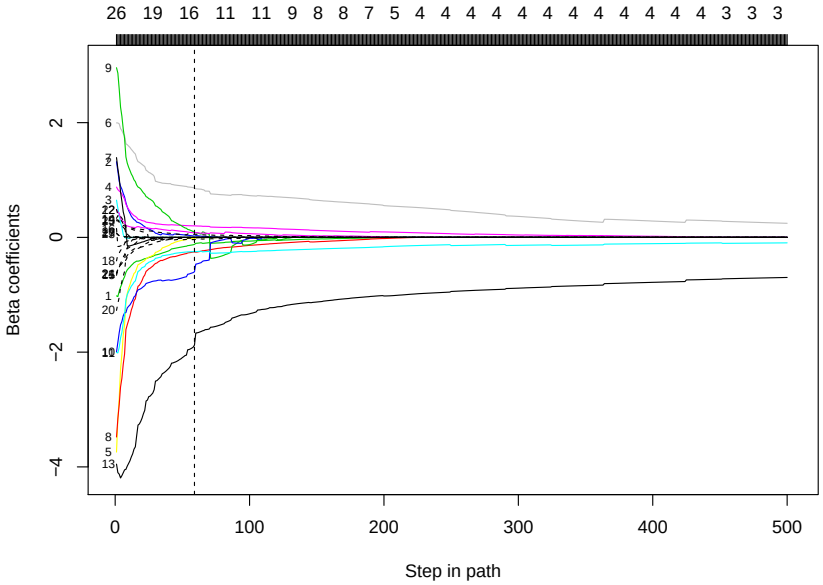

Figure 3. *Path of estimates of β as λ is increased from zero. The broken lines correspond to the noise variables 14–26. The number of non-zero coefficients is indicated along the top. The vertical broken line is drawn at the model having approximately minimum test error.*

(a) Compute the residual $\hat{r}$ from the Lasso solution

(b) Fit a neural network to $\hat{r}$ using just the features with non-zero weights in the Lasso solution, giving $\hat{z}$.

(c) Construct the final solution as $\hat{r} + \hat{z}$

All neural nets in the study had $K = 50$ hidden units and used the sigmoid transfer function; the bound parameter M for `LassoNet` was set to 1. We created a training set of size $n = 400, p = 20$, with 6 different scenarios of the form $y = x\beta_0 + g(x) + \sigma \cdot Z$, with $\sim N(0, 1)$. For the 6th scenario, $n = 1000, p = 200$. We also generated a validation set of size n and a test set of size 1000 from the same population. The validation set was used to choose the tuning parameter (λ for Lasso, two-stage and `LassoNet`, number of training epochs for the NN) and then chosen model was evaluated on the test set.

1. Linear $x\beta_0$; $\beta_0 = (\pm 1, \pm 1, \pm 1, 0, 0, \ldots 0;), \sigma = 1$, with signs chosen at random
2. Linear + Nonlinear, with strong linear signal: $\beta_0 = (\pm 6, \pm 6, \pm 6, 0, 0, \ldots 0)$; $\sigma = 8$, $g(x) = 2(x_1^3 - 3x_1) + 4(x_2^2 x_3 - x_3)$;
3. Linear + Nonlinear, with weak linear signal: $\beta_0 = (\pm 1, \pm 1, \pm 1, 0, 0, \ldots 0)$; $\sigma = 4$, $g(x) = 2(x_1^3 - 3x_1) + 4(x_2^2 x_3 - x_3)$;
4. Linear + Nonlinear, with weak linear signal and non-hierarchical structure: $\beta_0 = (0, 0, 0, \ldots, \pm 1, \pm 1, \pm 1)$; $g(x) = 2(x_1^3 - 3x_1) + 4(x_2^2 x_3 - x_3), \sigma = 4$.
5. Sum of two sigmoid function $s(x)$ in all features: $\beta_0 = 0, g(x) = s(b_1^T x) + s(b_2^T x); b_1, b_2 \sim N(2, 1), \sigma = 0.5$.
6. Friedman's function: $\beta_0 = 0$, $g(x) = 9 \prod_{j=1}^{3} (\exp(-3(1-x_j)^2)) - 0.8 \exp(-2(x_4 - x_5) + 2(\sin(\pi x_6)^2) - 2.5x_7 - x_8)$ This is taken from Friedman (1991), where it was used to assess the MARS procedure.

All features were generated as independent standard Gaussians, except in scenario 6 where each entry was chosen uniformly from $(0, 1, 2, \ldots 9)/10$. In each of settings 2–4, the nonlinear signal $g(x)$ is uncorrelated with the linear component. Figure 4 shows the test set mean squared errors over 10 realizations for each method, in the six scenarios.

In the top left (linear model setting), Lasso, two-stage and `LassoNet` perform well, with the neural network lagging behind. `LassoNet` shows an advantage in scenarios 2-4, where there the underlying function has both linear and nonlinear components. The non-hierarchical setup has only a mild effect on `LassoNet`.

Scenario 5 is a setup for the neural network, and non-surprisingly, it performs best. In Scenarios 6 we see that `LassoNet` and the two stage procedure outperform the lasso, and provide considerable improvement over a standard neural network.

5 Real data examples

In looking for real data examples, we naturally sought problems where a standard neural network outperformed a linear model fit via the Lasso. Since are are not working with deep (multi hidden layer) nets in this paper, we didn't consider problems where multi-level feature extraction and convolution can be effective. Rather we considered "flat" learning problems, where a single hidden layer net would be effective. And we also focussed on observational data where the SNR is relatively low, and feature interpretation is likely to be important. To our surprise, we found that for most the problems we looked at (e.g. from the UCI database), the Lasso performed as well as the Neural Net. We wonder what experience others have had in this regard.

5.1 NRTI HIV data

Rhee et al. (2003) study six nucleoside reverse transcriptase inhibitors (NRTIs) that are used to treat HIV-1. The target of these drugs can become resistant through mutation, and they compare a collection of models for predicting these drug's (log) susceptibility, a measure of drug resistance based on the location of mutations. We used the data on the first inhibitor, for which there are $p = 217$ sites and $n = 1073$ samples, and divided the data into roughly equal-sized training and test sets. The test set mean squared errors are shown in Figure 5. `LassoNet` improves upon both the lasso and a neural network, and does best using only about 60 features. Figure 6 shows the number of features in common between Lasso and `LassoNet` fits, as we move along the solution paths. We see nearly 100% agreement, which is reassuring and aids in the interpretation of the `LassoNet` model.

5.2 MNIST data

We selected a training set of 1000 1s and 2s from the well known MNIST dataset, each image being a 28×28 matrix of gray scale values from 0 to 255. In this problem a multi-layer deep net is likely to work best, but for illustration here we consider here just single layer net. We applied the lasso, a single hidden layer NN with $K = 50$ hidden units, and the `LassoNet` with the same architecture and $M = 1$. Figure 7 show the results from lasso, NN and `LassoNet`. For the standard NN we have plotted the 28×28 image with entries $m_j = \sum_k |\hat{\theta}_{jk}|$ as a measure of the importance of each feature j. The `LassoNet` solution represents the one with about 60 non-zero features, and uncovers the pixels most important for disciminating 1s from 2s.

5.3 Ames housing dataset

This example is taken from Cock (2011) and was used in a Kaggle competition "House Prices: Advanced Regression Techniques'. The goal is to predict housing prices in Ames, Iowa. There are 1460 observations, and 863 predictors after one-hot encoding the categorial predictors and removing predictors with missing values. We divided the data into training and test sets, in proportions $(2/3, 1/3)$.

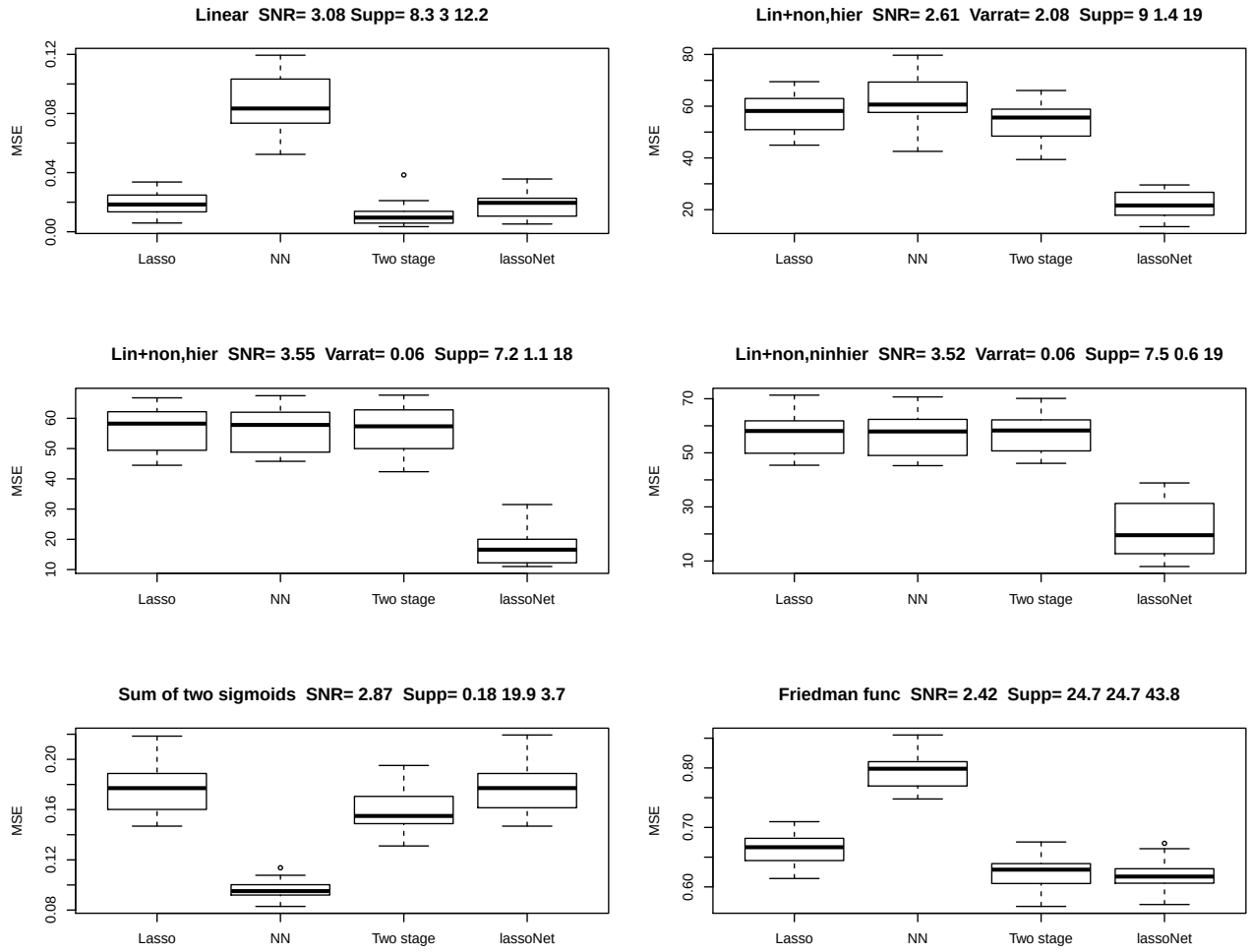

Figure 4. *Results of the simulation study, comparing Lasso, single hidden layer neural network, a neural network with a linear term fit by a two stage process, and* `LassoNet`*. Above each panel, the scenario is summarized with the signal-to-noise ratio SNR, "Varrat" — the ratio between the variances of the linear and non-linear parts of the underlying function, and "Supp"— the average number of non-zero feature weights for the Lasso, Two-stage and* `LassoNet` *procedures.*

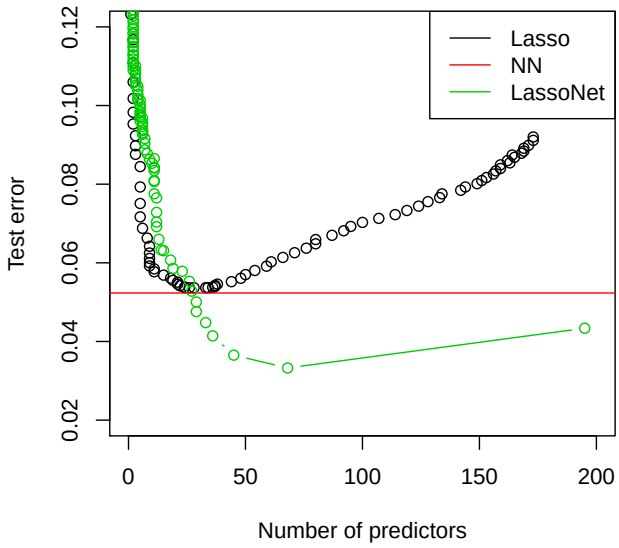

Figure 5: *Test set errors for NRTI example.*

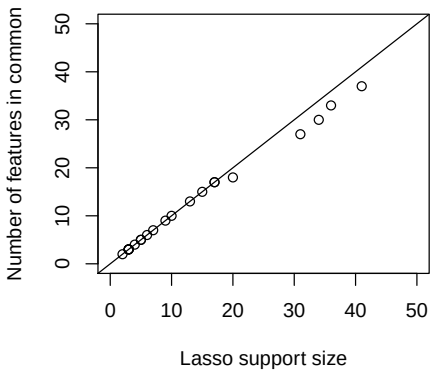

Figure 6: *NRTI example: number of features in common between Lasso and* `LassoNet` *fits.*

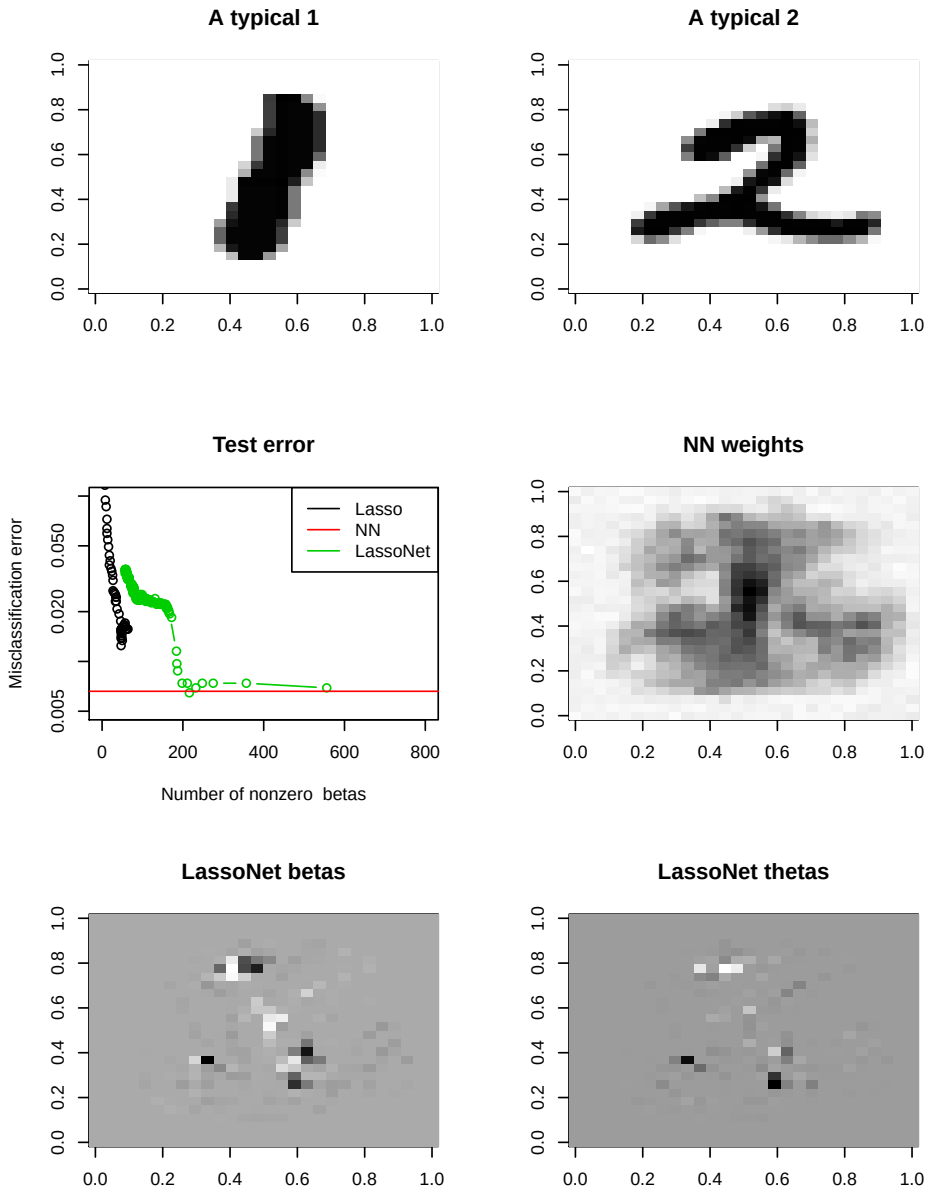

Figure 7. *MNIST example, discriminating 1s from 2s. Shown are typical digits, the test error from lasso, NN and* `LassoNet` *and estimated weights from the latter two methods.*

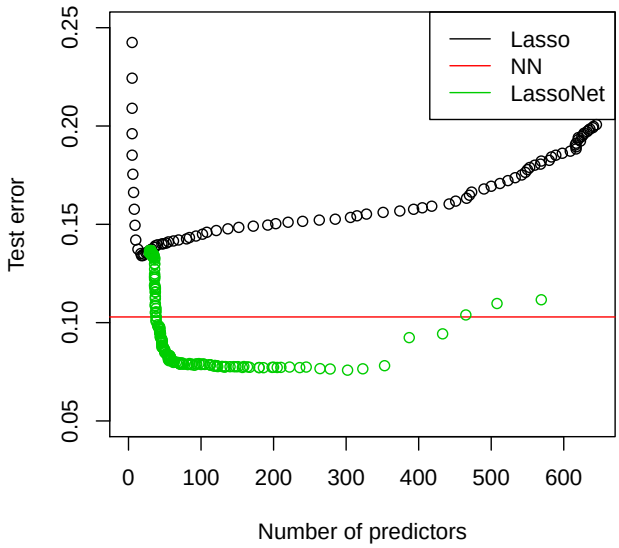

Figure 8: *Ames housing data: test set errors for competing methods.*

Figure 8 shows the test set prediction errors. We see that `LassoNet` offers a substantial improvement in test error, using only a fraction of the total predictors.

6 Optimization formulation

Our optimization problem consists in minimizing the objective function defined in (2):

$$\begin{aligned} \underset{\alpha,\beta,\gamma,\Theta}{\text{minimize}} \quad & \frac{1}{2}\sum_i (y_i - \hat{y}_i(\alpha,\beta,\gamma,\Theta))^2 + \lambda \sum_j |\beta_j|, \\ \text{subject to} \quad & |\theta_{j,k}| \le M \cdot |\beta_j| \quad \forall j,k. \end{aligned} \tag{4}$$

where our prediction is defined by

$$\hat{y}_i = \hat{y}_i(\alpha,\beta,\gamma,\Theta) = \hat{\beta}_0 + \sum_j x_{i,j}\hat{\beta}_j + \sum_{k=1}^{K}[\hat{\alpha}_k + \hat{\gamma}_k \cdot f(\theta_k^T x_i)],$$

for each $i \in [n]$. For simplicity we have omitted the γ penalty in (2). Unfortunately, this problem is non-convex due to the presence of the nonlinear activation function f, and to the inequality constraint. However, it can be tackled using the proximal gradient descent algorithm [Parikh et al. (2014), Beck (2017)]. For notation shorthand, we introduce

$$\mathcal{L}(\alpha,\beta,\gamma,\Theta) = \frac{1}{2}\sum_i (y_i - \hat{y}_i(\alpha,\beta,\gamma,\Theta))^2.$$

The proximal gradient descent algorithm is an iterative procedure and we use $(\alpha^{(t)},\beta^{(t)},\gamma^{(t)},\Theta^{(t)})$ to denote the parameters, and δ_t to denote the stepsize at the t-th iteration. The algorithm updates

$(\alpha, \beta, \gamma, \Theta)$ via the following:

$$
\begin{aligned}
&(\alpha^{(t+1)}, \beta^{(t+1)}, \gamma^{(t+1)}, \Theta^{(t+1)}) \\
&= \underset{(\alpha,\beta,\gamma,\Theta):\ |\theta_{j,k}| \le M \cdot |\beta_j|\ \forall j,k}{\operatorname{argmin}} \langle \nabla_\alpha \mathcal{L}, \alpha - \alpha^{(t)} \rangle + \langle \nabla_\beta \mathcal{L}, \beta - \beta^{(t)} \rangle + \langle \nabla_\gamma \mathcal{L}, \gamma - \gamma^{(t)} \rangle + \langle \nabla_\Theta \mathcal{L}, \Theta - \Theta^{(t)} \rangle \\
&\qquad + \frac{1}{2\delta_t} \left(\left\| \alpha - \alpha^{(t)} \right\|_2^2 + \left\| \beta - \beta^{(t)} \right\|_2^2 + \left\| \gamma - \gamma^{(t)} \right\|_2^2 + \| \Theta - \Theta_t \|_F^2 \right)
\end{aligned} \tag{5}
$$

It is easy to find the update rule for $(\alpha^{(t+1)}, \beta_0^{(t+1)}, \gamma^{(t+1)})$. In fact, Eq (5) shows that

$$
\begin{aligned}
\alpha^{(t+1)} &= \alpha^{(t)} - \delta_t \nabla_\alpha \mathcal{L}(\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)}, \Theta^{(t)}) \\
\beta_0^{(t+1)} &= \beta_0^{(t)} - \delta_t \partial_{\beta_0} \mathcal{L}(\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)}, \Theta^{(t)}) \\
\gamma^{(t+1)} &= \gamma^{(t)} - \delta_t \nabla_\gamma \mathcal{L}(\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)}, \Theta^{(t)})
\end{aligned}
$$

Moreover, Eq (5) implies that $(\beta_{[1:p]}^{(t+1)}, \Theta^{(t+1)})$ satisfies the following: for each $1 \le j \le p$,

$$
\begin{aligned}
(\beta_j^{(t+1)}, \Theta_{j,\cdot}^{(t+1)}) = \underset{(\beta_j, \Theta_{j,\cdot}): \|\Theta_{j,\cdot}\|_\infty \le |\beta_j|}{\operatorname{argmin}} & (\beta_j - (\beta_j^{(t)} - \delta_t \cdot \partial_{\beta_j} \mathcal{L}(\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)}, \Theta^{(t)}))^2 \\
& + \left\| \Theta_{j,\cdot} - (\Theta_{j,\cdot}^{(t)} - \delta_t \cdot \partial_{\Theta_{j,\cdot}} \mathcal{L}(\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)}, \Theta^{(t)})) \right\|_2^2
\end{aligned} \tag{6}
$$

Now, for each $1 \le j \le p$, $(\beta_j^{(t+1)}, \Theta_j^{(t+1)})$ is the solution to the minimization problem defined as right-hand side of Eq (6). Despite the problem's non-convexity, we are able to develop an efficient iterative algorithm. Our approach relies crucially on the proposition below, whose proof is deferred to Appendix A.

Proposition 1. *Fix $u \in \mathbb{R}^p$, $v \in \mathbb{R}$ and $\lambda, M \in \mathbb{R}_{++}$. Consider the optimization problem:*

$$
\begin{aligned}
&\underset{\beta,\theta}{\text{minimize}} \quad L(\beta, \theta) \equiv \frac{1}{2}(v - \beta)^2 + \frac{1}{2} \|u - \theta\|_2^2 + \lambda |\beta|. \\
&\text{subject to} \quad \|\theta\|_\infty \le M |\beta|.
\end{aligned} \tag{7}
$$

A sufficient and necessary condition for (β^*, θ^*) to be the optimum of problem (Eq (7)) is the following: there exist some $w \in \mathbb{R}_+$ and $N \in \{1, 2, \ldots, p\}$ such that

(i)

$$
w = \frac{1}{N + \frac{1}{M^2}} \Big(\frac{1}{M}(|v| - \lambda) + \sum_{i \in [N]} |u_{(i)}| \Big) \in \big[|u_{(N+1)}|, |u_{(N)}| \big) \tag{8}
$$

where $|u_{(1)}| \ge |u_{(2)}| \ge \ldots \ge u_{|(p)|} \ge 0$ are the order statistics of the coordinates of $|u| \in \mathbb{R}^p$.

(ii)

$$
\beta^* = \operatorname{sign}(v) w \quad \textit{and} \quad \theta_i^* = \operatorname{sign}(u_i) \min\{w, |u_i|\} \quad \textit{for } i \in [p]
$$

Proposition 1 naturally leads to an efficient method for solving the update in Eq (6), given Algorithm 1 below.

7 Extensions of the procedure and discussion

There a number of ways that the `LassoNet` procedure can be generalized. These include:

Algorithm 1 Subroutine to update $(\beta_{[1:p]}^{(t+1)}, \Theta^{(t+1)})$

1: Input $\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)}, \Theta^{(t)}$ and stepsize δ.
2: **for** $j = 1, 2, \ldots, p$ **do**
3: Compute $v_j^{(t)} \in \mathbb{R}, u_j^{(t)} \in \mathbb{R}^K$

$$v_j^{(t)} = \beta_j^{(t)} - \delta \cdot \partial_{\beta_j} \mathcal{L}(\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)}, \Theta^{(t)})$$
$$u_j^{(t)} = \Theta_{j,\cdot}^{(t)} - \delta \cdot \nabla_{\Theta_{j,\cdot}} \mathcal{L}(\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)}, \Theta^{(t)})$$

4: Compute the order statistics of $|u|$:

$$|u_{(1)}| \geq |u_{(2)}| \geq \ldots \geq u_{|(K)|} \geq 0.$$

5: Find $m \in \{0, 1, \ldots, K\}$ such that

$$w_m = \frac{1}{m + \frac{1}{M^2}} \Big(\frac{1}{M}(|v| - \lambda) + \sum_{i \in [m]} |u_{(i)}| \Big) \in \big[|u_{(m+1)}|, |u_{(m)}| \big)$$

where by convention we define $u_{K+1} = 0$ and $u_0 = \infty$.
6: Update the parameters

$$\beta_j^{(t+1)} = \frac{1}{M} \operatorname{sign}(v_j^{(t)}) w_m \quad \text{and} \quad \Theta_{j,k}^{(t+1)} = \operatorname{sign}(u_k^{(t)}) \min\{w_m, |u_k^{(t)}|)\} \quad \text{for } k \in [K].$$

7: **end for**

- *Multi-layer (deep) networks.* One can add additional hidden layers, without any change to the constraints. Since the raw features are modelled only in the first hidden layer, feature sparsity will be attained in the same manner.
- *Non-linear sparse principal components and auto-encoders.* Here we predict $X_1, X_2, \ldots X_p$ from itself, and so have p connections in the final layer.

These will be topics of future research.

Acknowledgements: We'd like to thank John Duchi and Ryan Tibshirani for helpful comments. Robert Tibshirani was supported by NIH grant 5R01 EB001988-16 and NSF grant 19 DMS1208164. A R-language package for `LassoNet` will be made freely available.

References

Beck, A. (2017), *First-order methods in optimization*, Vol. 25, SIAM.

Cock, D. D. (2011), 'Ames, iowa: Alternative to the boston housing data as an end of semester regression project', *Journal of Statistics Education* **19**(3), null.
URL: *https://doi.org/10.1080/10691898.2011.11889627*

Friedman, J. (1991), 'Multivariate adaptive regression splines (with discussion)', *Annals of Statistics* **19**(1), 1–141.

Friedman, J., Hastie, T. & Tibshirani, R. (2010), 'Regularization paths for generalized linear models via coordinate descent', *Journal of Statistical Software* **33**, 1–22.

Guyon, I. & Elisseeff, A. (2003), 'An introduction to variable and feature selection', *Journal of Machine Learning Research* **55**, 1157–1182.

Han, S., Pool, J., Tran, J. & Dally, W. J. (2015), 'Learning both weights and connections for efficient neural network', *Advances in Neural Information Processing Systems* **55**, 11351143.

Parikh, N., Boyd, S. et al. (2014), 'Proximal algorithms', *Foundations and Trends® in Optimization* **1**(3), 127–239.

Reed, R. (1993), 'Pruning algorithms - a survey', *IEEE Transactions on Neural Networks* **4**, 740–747.

Rhee, S.-Y., Gonzales, M. J., Kantor, R., Betts, B. J., Ravela, J. & Shafer, R. W. (2003), 'Human immunodeficiency virus reverse transcriptase and pro- tease sequence database', *Nucleic Acids Research* **31**, 298–303.

Scardapane, S., Hussain, A., Uncini, A. & Comminiello, D. (2017), 'Group sparse regularizations for deep neural networks', *Journal, Neurocomputing, Elsevier Science Publishers* **241**, 8189.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society, Series B* **58**, 267–288.

Verikas, A. & Bacauskiene, M. (2002), 'Feature selection with neural networks', *Pattern Recognition Letters* **23**(11), 1323–1335.

Webber, M., Striakus, J., Schumacher, M. & Binder, H. (2018), Network-constrained covariate coefficient and connection sign estimation, Technical report, CORE Discussion Paper 2018/18 -OR- Bank of Lithuania Discussion Paper.

Wen, W., Wu, C., Wang, Y., Chen, Y. & Li, H. (2016), Learning structured sparsity in deep neural networks, *in* 'Advances in Neural Information Processing Systems', pp. 1–10.

Yoon, J. & Hwang, S. J. (2017), Combined group and exclusive sparsity for deep neural networks, *in* 'Proceedings of the 34th International Conference on Machine Learning-Volume 70', JMLR.org, pp. 3958–3966.

A Proof of Proposition 1

We start by proving the claim below: for some $w \in \mathbb{R}_+$

$$\text{Claim}: \ M\beta^* = \text{sign}(v)w \ \text{ and } \ \theta_i^* = \text{sign}(u_i)\min\{w, |u_i|\} \ \text{ for } i \in [p]. \tag{9}$$

Indeed, note first that $\text{sign}(\beta^*) = \text{sign}(v)$, since otherwise $(-\beta^*, \theta^*)$ achieves a strictly smaller objective than (β^*, θ^*). Now, we denote $w = M|\beta^*|$. Certainly, we have

$$M\beta^* = \text{sign}(v)w. \tag{10}$$

Moreover, by definition, θ^* is the minimum for the optimization problem below:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & L(\beta^*, \theta) = \frac{1}{2}(v - \beta^*)^2 + \lambda|\beta^*| + \frac{1}{2}\|u - \theta\|_2^2 \\ \text{subject to} \quad & \|\theta\|_\infty \le M|\beta^*| = w \end{aligned}$$

This optimization problem is convex in θ. Since Slater's condition holds we have strong duality for the optimization problem. Hence, there exists some dual variable $s \in \mathbb{R}_+^p$ such that θ^* minimizes the Lagrangian function below:

$$\theta^* = \underset{\theta \in \mathbb{R}^p}{\text{argmin}} \, L(\beta^*, \theta) + \sum_{i=1}^p s_i(|\theta_i| - w) = \underset{\theta \in \mathbb{R}^p}{\text{argmin}} \, \frac{1}{2}\|u - \theta\|_2^2 + \sum_{i=1}^p s_i|\theta_i|.$$

Now, we take the subgradient and get the characterization below: for each $i \in [p]$,

$$\begin{aligned} & \theta_i - u_i + s_i v_i = 0 \ \text{ for some } v_i \in \partial(|\theta_i|). \\ & s_i(|\theta_i| - w) = 0. \\ & s_i \ge 0, \ \text{ and } \ |\theta_i| \le w. \end{aligned} \tag{11}$$

Now we divide our discussion into two cases:

1. $s_i = 0$. By the KKT condition (Eq. (11)), we get $u_i = \theta_i$. By the KKT condition Eq. (11)) again, that $u_i = \theta_i$ could be a possible solution if and only if $|u_i| \le w$.
2. $s_i > 0$. By the KKT condition (Eq. (11)), we get that $|\theta_i| = w$. Now, the fact that $u_i = \theta_i + s_i v_i$ for some $v_i \in \partial(|\theta_i|)$ implies that $\text{sign}(\theta_i) = \text{sign}(u_i)$. Hence $\theta_i = \text{sign}(u_i)w$. Note that, if $w \neq 0$, we must have $v_i = \text{sign}(\theta_i) = \text{sign}(u_i)$. Now, the fact that having some $s_i \ge 0$ satisfying $u_i = \theta_i + s_i v_i = \text{sign}(u_i)(w + s_i)$ is equivalent to $|u_i| \ge w$.

Summarizing the above discussion, we see for each $i \in [p]$, θ_i^* must satisfy

$$\theta_i^* = \text{sign}(u_i)\min\{w, |u_i|)\} \tag{12}$$

Now, Eq (10) and Eq (12) together give the desired claim at Eq (9).

Now, back to the proof of the proposition. Define $\beta(w) : \mathbb{R}_+ \to \mathbb{R}$ and $\theta(w) : \mathbb{R}_+ \to \mathbb{R}^p$ by

$$\beta(w) = M^{-1}\,\text{sign}(v)w \ \text{ and } \ \theta_i(w) = \text{sign}(u_i)\min\{w, |u_i|\}.$$

The claim at Eq (9) allows us to reduce the original minimization problem (i.e., Eq (7)) to the problem that finds w that minimizes $L(\beta(w), \theta(w))$. Note that $L(\beta(w), \theta(w))$ is a piecewise smooth function. For each $w \in [|u_{(N+1)}|, |u_{(N)}|)$, we can compute $L(\beta(w), \theta(w))$ and get

$$L(\beta(w), \theta(w)) = \left(N + \frac{1}{M^2}\right)\left[\frac{1}{2}\left(w - \frac{\frac{1}{M}(|v| - \lambda) + \sum_{i \in [N]}|u_{(i)}|}{N + \frac{1}{M^2}}\right)^2\right] + C_N$$

where C_N is some remainder term that is independent of w (but can be dependent of v, u and N). Hence, $L(\beta(w), \theta(w))$ is smooth for $w \in (|u_{(N+1)}|, |u_{(N)}|)$. Now, define for each $N \in [p]$,

$$w_N = \frac{1}{\frac{1}{M^2} + N}\Big(\frac{1}{M}(|v| - \lambda) + \sum_{i \in [N]} |u_{(i)}|\Big). \tag{13}$$

Clearly, if $w_N \in [|u_{(N+1)}|, |u_{(N)}|)$, then w_N is a local minimum of $L(\beta(w), \theta(w))$ over $[|u_{(N+1)}|, |u_{(N)}|)$. Now we note the observation below:

$$\text{Observation: there exists a unique } N \in \{0, 1, 2 \ldots, p\} \text{ such that } w_N \in [|u_{(N+1)}|, |u_{(N)}|) \tag{14}$$

We defer the proof of this observation. An immediate consequence is that the minimum w^* for $L(\beta(w), \theta(w))$ over $w \in [0, \infty)$ is the unique w that satisfies $w = w_N \in [|u_{(N+1)}|, |u_{(N)}|)$.

Now, we prove the observation at Eq (14). Suppose for $N = N^*$, we have $w_N \in [|u_{(N+1)}|, |u_{(N)}|)$. Then

$$\begin{aligned} w_M &\geq |u_{(M)}| \quad \text{for } M > N^*. \\ w_M &\leq |u_{(M+1)}| \quad \text{for } M < N^*. \end{aligned} \tag{15}$$

which justifies the uniqueness of N satisfying $w_N \in [|u_{(N+1)}|, |u_{(N)}|)$. We first show that $w_M \geq |u_{(M)}|$ for all $M > N^*$. By definition of N^*, $w_{N^*} \geq |u_{(M)}|$, which, by definition of w_{N^*}, is equivalent to

$$\sum_{i \in [N]} (|u_{(i)}| - |u_{(M)}|) \geq \frac{1}{M^2}|u_{(M+1)}| - \frac{1}{M}(|v| - \lambda).$$

Since $M > N$, this also implies that

$$\sum_{i \in [M]} (|u_{(i)}| - |u_{(M)}|) \geq \frac{1}{M^2}|u_{(M+1)}| - \frac{1}{M}(|v| - \lambda).$$

Thus, it gives $w_M \geq |u_{(M)}|$. This proves that $w_M \geq |u_{(M)}|$ for all $M > N^*$. Similarly, one can prove $w_M \leq |u_{(M+1)}|$ for all $M < N^*$, and we omit the proof of this part.

References

Beck, A. (2017), *First-order methods in optimization*, Vol. 25, SIAM.

Cock, D. D. (2011), 'Ames, iowa: Alternative to the boston housing data as an end of semester regression project', *Journal of Statistics Education* **19**(3), null.
URL: *https://doi.org/10.1080/10691898.2011.11889627*

Friedman, J. (1991), 'Multivariate adaptive regression splines (with discussion)', *Annals of Statistics* **19**(1), 1–141.

Friedman, J., Hastie, T. & Tibshirani, R. (2010), 'Regularization paths for generalized linear models via coordinate descent', *Journal of Statistical Software* **33**, 1–22.

Guyon, I. & Elisseeff, A. (2003), 'An introduction to variable and feature selection', *Journal of Machine Learning Research* **55**, 1157–1182.

Han, S., Pool, J., Tran, J. & Dally, W. J. (2015), 'Learning both weights and connections for efficient neural network', *Advances in Neural Information Processing Systems* **55**, 11351143.

Parikh, N., Boyd, S. et al. (2014), 'Proximal algorithms', *Foundations and Trends® in Optimization* **1**(3), 127–239.

Reed, R. (1993), 'Pruning algorithms - a survey', *IEEE Transactions on Neural Networks* **4**, 740–747.

Rhee, S.-Y., Gonzales, M. J., Kantor, R., Betts, B. J., Ravela, J. & Shafer, R. W. (2003), 'Human immunodeficiency virus reverse transcriptase and pro- tease sequence database', *Nucleic Acids Research* **31**, 298–303.

Scardapane, S., Hussain, A., Uncini, A. & Comminiello, D. (2017), 'Group sparse regularizations for deep neural networks', *Journal, Neurocomputing, Elsevier Science Publishers* **241**, 8189.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society, Series B* **58**, 267–288.

Verikas, A. & Bacauskiene, M. (2002), 'Feature selection with neural networks', *Pattern Recognition Letters* **23**(11), 1323–1335.

Webber, M., Striakus, J., Schumacher, M. & Binder, H. (2018), Network-constrained covariate coefficient and connection sign estimation, Technical report, CORE Discussion Paper 2018/18 -OR- Bank of Lithuania Discussion Paper.

Wen, W., Wu, C., Wang, Y., Chen, Y. & Li, H. (2016), Learning structured sparsity in deep neural networks, *in* 'Advances in Neural Information Processing Systems', pp. 1–10.

Yoon, J. & Hwang, S. J. (2017), Combined group and exclusive sparsity for deep neural networks, *in* 'Proceedings of the 34th International Conference on Machine Learning-Volume 70', JMLR.org, pp. 3958–3966.