Sparse-Input Neural Networks for High-dimensional Nonparametric Regression

Jean Feng [1] **Noah Simon** [1]

Abstract

Neural networks are usually not the tool of choice for nonparametric high-dimensional problems where the number of input features is much larger than the number of observations. Though neural networks can approximate complex multivariate functions, they generally require a large number of training observations to obtain reasonable fits, unless one can learn the appropriate network structure. In this manuscript, we show that neural networks can be applied successfully to high-dimensional settings if the true regression function falls in a low dimensional subspace, and proper regularization is used. We propose fitting a neural network with a sparse group lasso penalty on the first-layer input weights, which results in a neural net that only uses a small subset of the original features. In addition, we characterize the statistical convergence of the penalized empirical risk minimizer to the true regression function: we show that the predictive error of this penalized estimator only grows with the logarithm of the number of input features; and we show that the weights of irrelevant features converge to zero. Via simulation studies and an analysis of a genomic dataset for the production of riboflavin, we show that these sparse-input neural networks outperform existing nonparametric high-dimensional estimation methods when the data has complex higher-order interactions.

1. Introduction

It is often of interest to predict a response y from a set of inputs x. In many applications, the relationship between x and y can be quite complex and non-linear. For low and moderate dimensional problems, there are many methods that have been effective for estimating complex non-linear relationships (e.g. kernel regression, aggregated regression trees, neural networks; Nadaraya (1964); Watson (1964); Breiman et al. (1984); Barron (1994)). Among these, neural networks have been particularly effective, building highly predictive models in complex domains where other methods have had limited success (e.g. speech recognition, computer vision, and natural language processing, among others; Graves et al. (2013); Krizhevsky et al. (2012); Szegedy et al. (2015); Socher et al. (2013); Mikolov et al. (2013)).

With the latest technological developments in biology and other fields, it is now very common to encounter high-dimensional data, where the number of features p may far exceed the number of observations n. For general problems in this setting, where the appropriate network structure is unknown, neural networks are rarely used since the number of training samples required for good performance is prohibitive. Instead, the popular methods in this setting include the simple Lasso (Tibshirani, 1996) and its nonparametric extensions such as Sparse Additive Models (SpAM) (Ravikumar et al., 2007) and high-dimensional additive models (Meier et al., 2009). These methods typically model the data as the sum of a small number of univariate (or very low-dimensional) functions. Unfortunately in data, the response may depend on complex interactions between multiple covariates, and failing to model these interactions can result in highly biased estimates.

In this paper, we propose an extension to neural networks that can both a) select a small subset of informative features for estimating the response, and b) using that collection, fit a potentially complex, non-linear response surface. Our method, sparse-input neural networks, fits a neural network with a mixed ℓ_1/ℓ_2 penalty, known as the sparse group lasso penalty (Simon et al., 2013). Here groups correspond to the weights connected to the same input node. The sparse group lasso penalty encourages fitting models with a sparse set of input features. This model can be trained using a generalized gradient descent algorithm.

To justify the use of this method, we provide theoretical and empirical evidence. First, we prove oracle inequalities that give probabilistic performance guarantees of our esti-

[1]University of Washington. Correspondence to: Jean Feng <jeanfeng@u.washington.edu>.

Presented at the ICML 2017 Workshop on Principled Approaches to Deep Learning, Sydney, Australia, 2017.

mator, assuming we have reached a global minimum of our penalized criterion. We show that if the response is best approximated by a sparse neural network that uses only s of the features, the prediction error of a sparse-input neural network shrinks at a rate of $O_p(n^{-1}s^{5/2}\log^2 n \log(p \vee n))$ (here we treat the number of hidden nodes as fixed). Hence the prediction error of sparse-input neural networks grows slowly with the number of features, making them suitable for high-dimensional problems. Moreover, we give an upper bound for the rate at which irrelevant neural network input-weights converge to zero. Though we only present theoretical results for the special case of the lasso, the proof techniques can be easily extended to the sparse group lasso. In addition, via simulation studies and a data analysis, we show that sparse-input neural networks can significantly outperform more traditional nonparametric high-dimensional methods when the true function is composed of complex higher-order interaction terms.

2. Related Work

A number of other authors have applied lasso and group lasso penalties to neural networks. That work has largely been focused on learning network structure however, rather than feature selection. Sun (1999) was one of the earliest papers that fit a neural network with a lasso penalty over all the weights. Scardapane et al. (2016) and Alvarez & Salzmann (2016) proposed using the sparse group lasso over all the weights in a deep neural network in order to learn a more compact network. The recent work by Bach (2017) is most closely aligned to our work: he considers convex neural networks with a single hidden layer, non-decreasing positively homogeneous activation functions, and an unbounded number of hidden nodes. He shows that theoretically, a lasso penalty on the input weights of such networks should perform well in high-dimensional settings.

The recent work in Zhang et al. (2016) examined the utility of regularization in deep learning. The authors found that regularization was not necessary for a neural network to have good performance and that using an appropriate network architecture led to larger decreases in generalization error. Our results support this claim since we use the sparse group lasso to learn the structure of the first layer.

Previously, statistical convergence rates for neural networks have been established when the ℓ_1-norm of the weights are constrained (Bartlett, 1998; Anthony & Bartlett, 2009). Here we consider the problem in its penalized form and take advantage of a sparsity assumption to improve the convergence rates. Bach (2017) show that the estimation error of their neural networks converges at a slow rate of $O_p(n^{-1/2}\sqrt{\log p})$. In our paper, we recover fast rates with n^{-1} instead of $n^{-1/2}$. In addition, we provide convergence rates for the weights connected to the irrelevant input features.

Our proofs for statistical convergence rates are inspired by Städler et al. (2010), which considers ℓ_1-penalization for mixture regression models. The techniques used in their paper are relevant as neural networks can be thought of as a mixture of regressions. Significant additional work was required however as the identifiability condition assumed in Städler et al. (2010) does not hold for neural networks.

3. Sparse-input neural networks

In this paper, we consider neural networks with a single hidden layer with m hidden nodes and a linear output. The neural network parameters are denoted $\boldsymbol{\eta} = (\boldsymbol{\theta}, \boldsymbol{t}, \boldsymbol{\beta}, b)$ where $\boldsymbol{\theta} \in \mathbb{R}^{p\times m}$ is the first-layer input weights, $\boldsymbol{t} \in \mathbb{R}^m$ is the intercept terms for the hidden nodes, $\boldsymbol{\beta} \in \mathbb{R}^m$ is the second-layer weights, and $b \in \mathbb{R}$ is an intercept term. Let $\boldsymbol{\theta}_j$ denote the weights tied to the jth hidden node and $\boldsymbol{\theta}_{(j)}$ denote the weights tied to the jth input feature. The activation function at each hidden node is a sigmoidal functions ψ, which means that ψ satisfies $\lim_{z\to-\infty}\psi(z) = 0$ and $\lim_{z\to\infty}\psi(z) = 1$. A neural network $f_{\boldsymbol{\eta}}$ maps input $\mathbf{x} \in \mathbb{R}^p$ to $\mathbb{R}$ as follows:

$$f_{\boldsymbol{\eta}}(\boldsymbol{x}) = \sum_{j=1}^{m} \beta_j \psi\left(\boldsymbol{\theta}_j^\top \boldsymbol{x} + t_j\right) + b. \tag{1}$$

Note that $\tanh$ neural networks are included in this framework since the $\tanh$ and sigmoid functions are related by a simple transformation.

Per the usual regression framework, suppose we have observations with response y_i and covariates $\boldsymbol{x}_i$ for $i = 1, ..., n$. We propose fitting a sparse-input neural network where we penalize the first-layer weights using a sparse group lasso penalty and the second-layer weights using a ridge penalty as follows:

$$\begin{aligned}\arg\min_{\eta\in\Theta} \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - f_{\boldsymbol{\eta}}(\boldsymbol{x}_i)\right)^2 + \lambda_0\|\boldsymbol{\beta}\|_2^2 \\ + \lambda_1\|\boldsymbol{\theta}\|_1 + \lambda_2\sum_{j=1}^{p}\|\boldsymbol{\theta}_{(j)}\|_2\end{aligned} \tag{2}$$

where $\Theta \subseteq \mathbb{R}^{mp+2m+1}$ is a convex set.

We have three penalties in this criterion. The ridge penalty on $\boldsymbol{\beta}$ serves to control the magnitude of the weights that are not in the first layer. The combination of the group lasso and lasso penalties is called the sparse group lasso (Simon et al., 2013). The group lasso penalty on $\boldsymbol{\theta}_{(j)}$ encourages sparsity at the input level. That is, it encourages the entire vector $\boldsymbol{\theta}_{(j)}$ to be zero. The lasso penalty on $\boldsymbol{\theta}$ encourages sparsity across all the weights, so the hidden nodes will be connected to only a few input nodes. Pictorially, (2) encourages fitting neural networks like that in Figure 1.

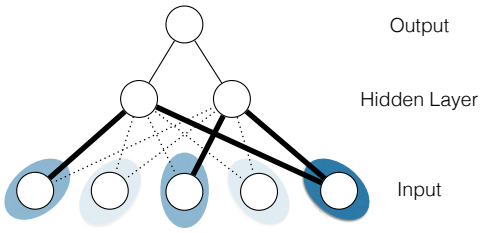


Figure 1. An example of a sparse-input neural network. The heavy and dotted lines indicate nonzero and zero weights, respectively. Each shaded oval corresponds to a group of first-layer weights. The weights from the dark blue oval are both nonzero. Each medium blue oval has a single nonzero weight, so they exhibit element-wise sparsity. All the weights in the light blue ovals are zero, so they exhibit group-level sparsity.

One could also consider adding a sparsity penalty to the second layer weights. This is useful if the neural network structure has a large number of hidden nodes that need to be pruned away. However in the high-dimensional setting, using a small number of hidden nodes generally works well.

3.1. Learning

Sparse-input neural networks can be trained using generalized gradient descent (Daubechies et al., 2004; Beck & Teboulle, 2009; Nesterov, 2004). Though generalized gradient descent was originally developed for convex problems, we can apply the recent work by Gong et al. (2013) to find a critical point in non-convex objective functions. Here we specialize their proposal, called GIST, to solve (2). This algorithm is similar to that in Alvarez & Salzmann (2016).

Let $\mathcal{L}_{x,y}^{\text{smooth}}(\eta)$ be the smooth component of the loss function in (2) as follows

$$\mathcal{L}_{x,y}^{\text{smooth}}(\boldsymbol{\eta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - f_{\boldsymbol{\eta}}(\boldsymbol{x}_i)\right)^2 + \lambda_0\|\boldsymbol{\beta}\|_2^2. \quad (3)$$

Let $S(\cdot,\cdot) : \mathbb{R}^p \times \mathbb{R} \mapsto \mathbb{R}^p$ be the coordinate-wise soft-thresholding operator

$$\left(S\left(\boldsymbol{z}, c\right)\right)_j = \text{sign}(z_j)\left(|z_j| - c\right)_+. \quad (4)$$

The algorithm for training a sparse-input neural network is given in Algorithm 1. The proximal gradient step is composed of three steps. The first step (6) performs a gradient update step only for the smooth component of the loss; the gradient can be computed using the standard back-propagation algorithm. The second and third steps, (7) and (8), are the proximal operations for the Sparse Group Lasso (Simon et al., 2013). It performs a soft-thresholding operation followed by a soft-scaling operation on each $\boldsymbol{\theta}_{(i)}$.

The initial step size at each iteration is chosen to be some value in $[\alpha_{min}, \alpha_{max}]$. In this paper, we simply used a fixed value, e.g. $\alpha_{min} = \alpha_{max}$, though one can also adaptively choose the initial step size (Barzilai & Borwein, 1988; Gong et al., 2013).

We choose the step size according to a monotone line search criterion, which accepts the step size α_k if the following condition is satisfied:

$$L(\boldsymbol{\eta}^{(k,2)}) \leq L(\boldsymbol{\eta}^{(k-1,2)}) - t\alpha_k\|\boldsymbol{\eta}^{(k,2)} - \boldsymbol{\eta}^{(k-1,2)}\|^2 \quad (5)$$

where $L(\cdot)$ is the objective function of (2) and $t \in (0,1)$. This line search criterion guarantees that Algorithm 1 will converge to a critical point (where the subdifferential contains zero) (Gong et al., 2013).

Finally, another option for training sparse-input neural networks is to apply the accelerated generalized gradient descent framework for non-convex objective functions developed in Ghadimi & Lan (2016). These are guaranteed to converge to a critical point, and have accelerated rate guarantees.

In short, these generalized gradient descent algorithms provide an efficient way to train sparse-input neural networks.

Algorithm 1 Training sparse-input neural networks

Initialize neural network parameters $\boldsymbol{\eta}^{(0,2)}$. Choose $s \in (0,1)$ and $\alpha_{min}, \alpha_{max}$ such that $\alpha_{max} \geq \alpha_{min} > 0$.
for iteration $k = 1, 2, \ldots$ **do**
  $\alpha_k \in [\alpha_{min}, \alpha_{max}]$
  repeat

$$\boldsymbol{\eta}^{(k,0)} = \boldsymbol{\eta}^{(k-1,2)} - \alpha_k \nabla_\eta \mathcal{L}_{x,y}^{\text{smooth}}(\boldsymbol{\eta}^{(k-1,2)}) \quad (6)$$

$$\boldsymbol{\theta}^{(k,1)} = S(\boldsymbol{\theta}^{(k,0)}, \alpha_k\lambda_1) \quad (7)$$

    for $i = 1, \ldots, p$ **do**

$$\boldsymbol{\theta}_{(i)}^{(k,2)} = \left(1 - \frac{\alpha_k\lambda_2}{\left\|\boldsymbol{\theta}_{(i)}^{(k,1)}\right\|_2}\right)_+ \boldsymbol{\theta}_{(i)}^{(k,1)} \quad (8)$$

    end for

$$\left(\boldsymbol{t}^{(k,2)}, \boldsymbol{\beta}^{(k,2)}, b^{(k,2)}\right) = \left(\boldsymbol{t}^{(k,0)}, \boldsymbol{\beta}^{(k,0)}, b^{(k,0)}\right)$$

  $\alpha_k \coloneqq s\alpha_k$
  until line search criterion is satisfied
end for

3.2. Tuning Hyper-parameters

There are four hyper-parameters for fitting a sparse-input neural network: three penalty parameters and the number of hidden nodes. The parameters should be tuned to ensure low generalization error of the model. Common methods for tuning hyper-parameters include cross-validation,

Nelder-Mead (Nelder & Mead, 1965), and Bayesian optimization methods (Snoek et al., 2012).

In this paper, we use cross-validation over a grid of hyperparameter values. In practice, the optimal number of hidden nodes for high-dimensional problems tends to be small, so only a few candidate values need be tested. Moreover, since there are only a small number of weights in the second layer of the neural network, the estimation error is quite robust to different ridge penalty parameter values. So we pre-tune the ridge penalty parameter and keep it fixed.

The most important parameters to tune are the lasso and group lasso penalties. We found that the lasso penalty can easily over-penalize the input weights and set everything to zero. Thus we only use candidate values where the lasso penalty parameter is no bigger than the group lasso penalty parameter.

4. Theoretical Guarantees

In this section, we provide probabilistic, finite-sample upper bounds on the prediction error of sparse-input neural networks. For this paper, we focus on the specific case where the input weights are penalized by the lasso and not the group lasso penalty. That is, we will be concerned with the following sparse-input neural network problem:

$$\hat{\boldsymbol{\eta}} = \arg\min_{\eta\in\Theta} \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - f_\eta(\boldsymbol{x}_i)\right)^2 + \lambda\|\boldsymbol{\theta}\|_1$$
$$\text{where } \Theta = \left\{\boldsymbol{\eta}\in\mathbb{R}^{mp+2m+1} : \|\boldsymbol{t}\|_2^2 + \|\boldsymbol{\beta}\|_2^2 + b^2 \le K\right\} \tag{9}$$

for a constant $K > 0$. Θ places a constraint on the norm of the second-layer weights, which is equivalent to using the ridge penalty. Though our results in this section are for the special case of the lasso, the proof techniques are quite general and can be extended to the group lasso penalty. All of our proofs are in the Supplementary Materials.

Notice that (9) assumes that the estimator is a global minimizer of a non-convex objective function. Since non-convex problems are typically computationally intractable to solve, there is admittedly a disconnect between the computational algorithm we have proposed and the theoretical results we establish in this section. Though it is desirable to establish theoretical properties for estimators arising from local optima, it is difficult to characterize their behavior and up to now, much of the theory for the Lasso depends on the estimator being a global minimizer. We do not address this issue and leave this problem for future research.

4.1. Problem Setup and Notation

Let $\mathbb{P}_{XY}$ be the joint distribution of the covariates X and response Y where X is sampled from $\mathcal{X} \subseteq [-X_{max}, X_{max}]^p$ and $\mathbb{P}_X$ be the marginal distribution over X. Given n observations, we denote the empirical distribution as $\mathbb{P}_n$.

Suppose the data is generated as the sum of a true function $f^* : \mathcal{X} \subseteq \mathbb{R}^p \mapsto \mathbb{R}$ and noise ϵ:

$$Y = f^*(X) + \epsilon$$

where ϵ is a sub-gaussian random variable with mean zero. That is, ϵ satisfies for some constants K_ϵ and σ_0,

$$K_\epsilon^2(\mathbb{E}e^{|\epsilon|^2/K_\epsilon^2} - 1) \le \sigma_0^2. \tag{10}$$

We suppose that ϵ is independent of X.

We use the neural network defined in (1) where the activation function is the sigmoid function $\psi(z) = 1/(1+e^{-z})$.

Next, we define a neural network equivalence class. Given parameter $\boldsymbol{\eta}$, the set of equivalent parameterizations are

$$EQ(\boldsymbol{\eta}) = \{\tilde{\boldsymbol{\eta}} \in \Theta : f_{\tilde{\boldsymbol{\eta}}}(x) = f_{\boldsymbol{\eta}}(x) \forall x \in \mathcal{X}\}. \tag{11}$$

By Lemma 1 in Supplementary Materials, the number of elements in $EQ(\boldsymbol{\eta})$ is finite if $\boldsymbol{\eta}$ does not have a hidden node with zero input weights (intercept term not included) and no two hidden nodes have the same exact input weights (modulo a sign flip). The proof for Lemma 1 is a straightforward extension of results in Albertini et al. (1993). In fact, $EQ(\boldsymbol{\eta})$ can only contain parameterizations that are sign-flips or rotations of each other.

We suppose the set of optimal neural networks that minimize the expected squared error loss is a single equivalence class

$$EQ_0 \equiv EQ(\boldsymbol{\eta}_0) = \arg\min_{\boldsymbol{\eta}\in\Theta} \mathbb{P}_X\|f^* - f_{\boldsymbol{\eta}}\|_2^2, \tag{12}$$

where $\boldsymbol{\eta}_0$ satisfies the conditions in Lemma 1 such that that EQ_0 is a finite set. In addition, suppose $\boldsymbol{\eta}_0$ has non-zero weights for features with indices $S \subseteq \{1, ..., p\}$ and zero weights for the remaining features S^c. We call S the set of relevant nodes since f^* is well-represented by just the nodes in S. Likewise, we call S^c the set of irrelevant nodes. Since the equivalence class only contains parameterizations that are rotations and sign-flips of one another, then every element in EQ_0 will have the same set of relevant nodes. Let $|S|$ denote the cardinality of S. In this work we are particularly interested in sparse optimal neural networks (with small $|S|$). Let the minimum distance to EQ_0 from any $\boldsymbol{\eta}$ be defined as

$$d_0(\boldsymbol{\eta}) = \min_{\boldsymbol{\eta}_0\in EQ_0} \|\boldsymbol{\eta} - \boldsymbol{\eta}_0\|_2$$

and the minimizer of $d_0(\boldsymbol{\eta})$ be denoted by

$$\boldsymbol{\eta}_0^{(\boldsymbol{\eta})} = (\boldsymbol{\theta}_0^{(\boldsymbol{\eta})}, \boldsymbol{t}_0^{(\boldsymbol{\eta})}, \boldsymbol{\beta}_0^{(\boldsymbol{\eta})}, b_0^{(\boldsymbol{\eta})}) \in \arg\min_{\boldsymbol{\eta}_0\in EQ_0} d_0(\boldsymbol{\eta}).$$

For notational shorthand, let the loss function be denoted $\ell_{\boldsymbol{\eta}}(y,x) = (y - f_{\boldsymbol{\eta}}(x))^2$. For the special case where $\boldsymbol{\eta}_0 \in EQ_0$, let $\ell_0(y,x) = (y - f_{\boldsymbol{\eta}_0}(x))^2$. Using this notation, the excess loss of a neural network $\boldsymbol{\eta}$ is defined as

$$\mathcal{E}(\eta) = \mathbb{P}_{XY}(\ell_{\boldsymbol{\eta}} - \ell_0). \tag{13}$$

Also let $\boldsymbol{\theta}_S$ denote the weights tied to the input nodes S and $\boldsymbol{\theta}_{S^c}$ denote the weights tied to the input nodes S^c.

4.2. Results

To understand the behavior of our estimated neural network from (9), we derive an upper bound on the combination of its excess loss and the ℓ_1-norm of irrelevant neural network weights. Our proof technique is inspired by Städler et al. (2010); however significant adaptations were needed to deal with the equivalence class of neural networks.

In order for our results to hold, we make the assumption that the expected loss is locally strongly convex at all $\boldsymbol{\eta}_0 \in EQ_0$. Since this only makes an assumption on the local behavior at EQ_0 and EQ_0 is a finite set, this assumption is relatively weak. More formally this assumption states:

Condition 1. *There is a constant $h_{min} > 0$ that depends on m, s, f^* and the distribution $\mathbb{P}_{XY}$ but does not depend on p such that for all $\boldsymbol{\eta}_0 \in EQ_0$,*

$$\left[\nabla^2_{\eta}\mathbb{P}\ell_{\eta}(\cdot)\right]_{\eta=\eta_0} \succeq h_{min}\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \tag{14}$$

where I is an $(m|S|+2m+1)\times(m|S|+2m+1)$ identity matrix and 0 are appropriately sized zero matrices. The top right corner of $\left[\nabla^2_{\eta}\mathbb{P}\ell_{\eta}(\cdot)\right]_{\eta=\eta_0}$ corresponds to the Hessian with respect to $(\boldsymbol{\theta}_S, \boldsymbol{t}, \boldsymbol{\beta}, b)$.

In addition, we need the following identifiability condition.

Condition 2. *For all $\epsilon > 0$, there is an $\alpha_\epsilon > 0$ that depends on m, s, f^* and the distribution $\mathbb{P}_{XY}$ but does not depend on p such that*

$$\alpha_\epsilon \leq \inf_{\boldsymbol{\eta}\in\Theta}\{\mathcal{E}(\boldsymbol{\eta}) : d_0(\boldsymbol{\eta}) \geq \epsilon \text{ and } 2\|\boldsymbol{\theta}_{S^c}\|_1 \leq \|\boldsymbol{\theta}_S - \boldsymbol{\theta}^{(\eta)}_{0,S}\|_1 + \|(\boldsymbol{t},\boldsymbol{\beta},b) - (\boldsymbol{t}^{(\eta)}_0, \boldsymbol{\beta}^{(\eta)}_0, b^{(\eta)}_0)\|_2\Big\}.$$

Condition 2 places a lower bound on the excess loss of neural networks outside the set of optimal neural networks EQ_0. However we only need this lower bound to apply to neural networks where the weight of the irrelevant nodes is dominated by the difference between $\boldsymbol{\eta}$ and $\boldsymbol{\eta}^{(\eta)}_0$ with respect $(\boldsymbol{\theta}_S, \boldsymbol{t}, \boldsymbol{\beta}, b)$. By restricting to this smaller set of neural networks, it is more realistic to claim that α_ϵ is independent of p.

With the local strong convexity and identifiability conditions, we have the following theorem. We use the notation $a \vee b = \max(a,b)$.

Theorem 1. *For any $\tilde{\lambda} > 0$ and $T \geq 1$, let*

$$\mathcal{T}_{\tilde{\lambda},T} = \Big\{\{(x_i,y_i)\}_{i=1}^n : \sup_{\eta\in\Theta}|(\mathbb{P}_n - \mathbb{P})(\ell_0 - \ell_\eta)| \leq T\tilde{\lambda}\left[\tilde{\lambda} \vee \left(\left\|(\boldsymbol{t},\boldsymbol{\beta},b) - \left(\boldsymbol{t}^{(\eta)}_0, \boldsymbol{\beta}^{(\eta)}_0, b^{(\eta)}_0\right)\right\|_2 + \|\boldsymbol{\theta} - \boldsymbol{\theta}^{(\eta)}_0\|_1\right)\right]\Big\}.$$

Suppose Conditions 1 and 2 hold. Suppose λ satisfies $3\tilde{\lambda}T \leq \lambda \leq a\tilde{\lambda}T$ for some constant $a \geq 3$. Let $\hat{\boldsymbol{\eta}}$ be the solution to (9). Then over the set $\mathcal{T}_{\tilde{\lambda},T}$, we have

$$\frac{1}{2}\mathcal{E}(\hat{\boldsymbol{\eta}}) + \left(\lambda - 2T\tilde{\lambda}\right)\|\hat{\boldsymbol{\theta}}_{S^c}\|_1 \leq \left(2T\tilde{\lambda} + \lambda\right)^2 \frac{m|S|C_0^2}{2} \tag{15}$$

where the constants are defined as follows:

$$C_0^2 = \frac{1}{\epsilon_0} \vee \frac{R^2}{\alpha_{\epsilon_0}} \tag{16}$$

$$\epsilon_0 = \frac{3h_{min}}{2C} \tag{17}$$

$$C = \frac{1}{48}G\left((3+a)\sqrt{m|S|} + 3\sqrt{2m+1}\right)^3 \tag{18}$$

$$R = 2K + (3+a)\max_{(\boldsymbol{\theta}_0,\cdot)\in EQ_0}\|\boldsymbol{\theta}_0\|_1 \tag{19}$$

$$G = \sup_{\eta\in\Theta}\max_{j_1,j_2,j_3\in\{1,\ldots,p\}}\left|\frac{\partial^3}{\partial\eta_{j_1}\partial\eta_{j_2}\partial\eta_{j_3}}\mathbb{P}\ell_\eta(\cdot)\right|. \tag{20}$$

Theorem 1 simultaneously bounds the excess loss and $\|\hat{\boldsymbol{\theta}}_{S^c}\|_1$. Consider the case where the identifiability constant α_{ϵ_0} is sufficiently large such that $C_0^2 = \epsilon_0^{-1}$. Then the above theorem states that the excess loss will be on the order of $O_p(\tilde{\lambda}^2 m^{5/2}|S|^{5/2})$ and the norm of $\hat{\boldsymbol{\theta}}_{S^c}$ will shrink at the rate of $O_p(\tilde{\lambda}m^{5/2}|S|^{5/2})$. If $\tilde{\lambda}$ shrinks as the number of samples increases, these values will go to zero. The convergence rate of the excess risk is faster for functions that are best approximated by neural networks that are more sparse (e.g. $|S|$ is small). The upper bound in (15) will also be small if $\tilde{\lambda}$ and m are small. However we must choose $\tilde{\lambda}$ carefully so that $\mathcal{T}_{\tilde{\lambda},T}$ occurs with high probability. The number of hidden nodes m also must be chosen carefully since m determines how well we can approximate f^* (Barron, 1994). This comes implicitly into our rate, as the "excess risk" is relative to the best neural net with m hidden nodes.

In the following theorem, we show that for a particular choice of $\tilde{\lambda}$, we can indeed ensure that $\mathcal{T}_{\tilde{\lambda},T}$ has high probability. The proof relies on techniques from empirical process theory.

Theorem 2. *Let*

$$\tilde{\lambda} = \frac{c_1}{\sqrt{n}}m^{3/2}\log(nm)\sqrt{\log(p \vee nm)}. \tag{21}$$

Then for any $T \geq 1$, we have

$$Pr_{X,Y}\left(\mathcal{T}_{\tilde{\lambda},T}\right) \geq 1 - c_2 \log n \exp\left(-\frac{T^2 m \log^2(nm)\log(p \vee nm)}{c_3}\right)$$

for constants $c_1, c_2, c_3 > 0$ that depend on X_{max}, K_ϵ, and σ_0.

If the identifiability constant is not too small, Theorems 1 and 2 state that if $\tilde{\lambda}$ is chosen according to (21), the excess loss converges at the rate

$$O_p\left(n^{-1} m^{11/2} |S|^{5/2} \log^2(nm)\log(p \vee nm)\right)$$

and the ℓ_1-norm of the irrelevant weights converges at the rate

$$O_p\left(n^{-1/2} m^4 |S|^{5/2} \log(nm)\sqrt{\log(p \vee nm)}\right).$$

Therefore the convergence rate of sparse-input neural networks should not drastically slow down with the number of features p since it only depends on its logarithm.

To the best of our knowledge, our results are the first to incorporate sparsity into the convergence rate of the norm of irrelevant weights. Though the convergence rates are likely not optimal, they help us understand why we observe better performance in sparse-input neural networks compared to standard neural networks in our simulation studies and data analysis. We hope to improve these convergence rate bounds in future research. In particular, we would like to shrink the exponent on m and $|S|$ since these rates become very slow for moderate values of m and $|S|$. In addition, our results depend on the constants h_{min} and α_{ϵ_0} in Conditions 1 and 2. In order for our rates to be useful, we would need to make sure that these constants do not shrink too quickly with m.

5. Simulation study

We now present a simulation study to understand how sparse-input neural networks compare against other nonparametric regression methods. We will consider the scenarios where the true function is the sum of univariate functions, a complex multivariate function, and a function that is in between these two extremes. In all the cases, the true function is sparse and only depends on the first six variables. We compare sparse-input neural networks to five methods, where the oracle methods are built using the first six variables:

- neural network with a single hidden layer and a ridge penalty on all the weights (ridge-only neural network);
- oracle ridge-only neural network;
- oracle additive univariate model of the form $\sum_{i=1}^{6} g_i(x_i)$ where g_i are fit using additive smoothing splines;
- oracle general multivariate model $g(x_1, ..., x_6)$ where g is fit using a 6-variate smoothing spline;
- Sparse Additive Model (SpAM), which fits an additive univariate model with a sparsity-inducing penalty (Ravikumar et al., 2007).

The oracle methods are not competitors in practice since they use information which will not be available; however, they give us some idea of how well our feature selection is working. Performance of the methods are assessed by the mean squared error (MSE) to the true function f^*, evaluated over randomly drawn covariates x in the test set.

5.1. Simulation Settings

For all of the simulations, we generated the data according to the model $y = f^*(x) + \sigma\epsilon$ where $\epsilon \sim N(0,1)$ and σ is scaled such that the signal to noise ratio is 2. We sampled the covariates $x \in \mathbb{R}^p$ from the standard uniform distribution. We generated a training set, a validation set, and a test set. The validation set was a quarter of the training set size and the test set was composed of 2000 observations. The penalty parameters in all the methods were tuned using the validation set. For neural networks, we also tuned the number of nodes in the hidden layer (5 or 10 nodes). We use $\psi = \tanh$ as the activation function in our neural networks. Each simulation was repeated 20 times.

5.2. Additive univariate model

In the first scenario, we have $p = 50$ covariates and the true function is the sum of univariate functions

$$f^*(x) = \sin(2x_1) + \cos(5x_2) + x_3^3 - \sin(x_4) + x_5 - x_6^2.$$

Since the true model is additive, we expect that the additive univariate oracle performs the best, followed by SpAM. As shown in Figure 2, we see that this is indeed the case.

We find that sparse-input neural networks also perform quite well. Thus if we are unsure if the true function is the sum of univariate functions, a sparse-input neural network can be a good option. In addition, we notice that the performance of sparse-input neural networks tends to track the oracle neural network and the multivariate oracle. In small sample sizes, sparse-input neural networks and oracle neural networks perform better than the multivariate oracle, as there is not enough data to support fitting a completely unstructured 6-variate smoother. As the number of samples increase, the multivariate oracle overtakes the sparse-input

neural network since it knows which features are truly relevant. We observe similar trends in the next two scenarios. The ridge-only neural network performs poorly in this scenario. Without a sparsity-inducing penalty, it is unable to determine which variables are relevant.

Figure 2 shows the norm of the first-layer weights in the sparse-input neural network, stratified by those connected to the relevant variables $x_1, ..., x_6$ and those connected to the remaining irrelevant variables. In this example the norm of the irrelevant weights appears to approach zero as the number of samples increases — this is in accordance with our theoretical results in Section 4.

5.3. Complex multivariate model

In the second simulation, we use $p = 50$ covariates and a sparse, multivariate regression function

$$f^*(x) = \sin(x_1(x_1 + x_2)) \cos(x_3 + x_4 x_5) \\ \sin(e^{x_5} + e^{x_6} - x_2).$$

Here we expect the general multivariate oracle to perform the best in large sample sizes, which is confirmed in Figure 2. Similar to results in Section 5.2, the performance of sparse-input neural networks nearly tracks the trajectory of oracle neural networks and the general multivariate oracle. As expected, the additive univariate oracle and SpAM perform very poorly. Their MSEs flatten out very quickly due to the bias from assuming an additive univariate model. In fact, we see that given a large enough training set, even a ridge-only neural network can outperform additive univariate oracle and SpAM.

Finally, we note that, as before, the norm of the irrelevant weights in the sparse-input neural network decreases toward zero as the number of samples increases.

5.4. High-dimensional additive multivariate model

Finally we consider a setting where we have a large number of input features, $p = 1000$. We use a regression function that is the sum of 3- and 4-variate functions:

$$f^*(x) = (x_1 \wedge x_2) \cos(1.5x_3 + 2x_4) + e^{x_5 + \sin(x_4)} x_2 \\ + \sin(x_6 \vee x_3))(x_5 - x_1).$$

This places it between the simple additive univariate function in the first scenario and the complex 6-variate function in the second scenario.

As shown in Figure 2, SpAM and sparse-input neural networks perform similarly in small samples and diverge at larger sample sizes. In particular, the sparse-input neural network starts outperforming SpAM as the oracle neural network and general multivariate oracle start outperforming the simple additive univariate oracle. This makes intuitive sense: at this point the irreducible bias from the additive univariate model begins to exceed the additional variance one has from estimating a 6-variate function. As before, the neural network trained with only a ridge penalty cannot take advantage of sparsity and performs poorly.

6. Analysis of riboflavin production

We now consider a high-throughput genomic data set for the production of riboflavin in *Bacillus subtilis* (Bühlmann et al., 2014). For each of the $n = 71$ experimental settings, we have gene expression profiles of $p = 4088$ genes as well as standardized log-transformed riboflavin production rates. Our goal is to predict the riboflavin production rate. We randomly split the dataset such that test set makes up one-fifth of the data and the rest are for training.

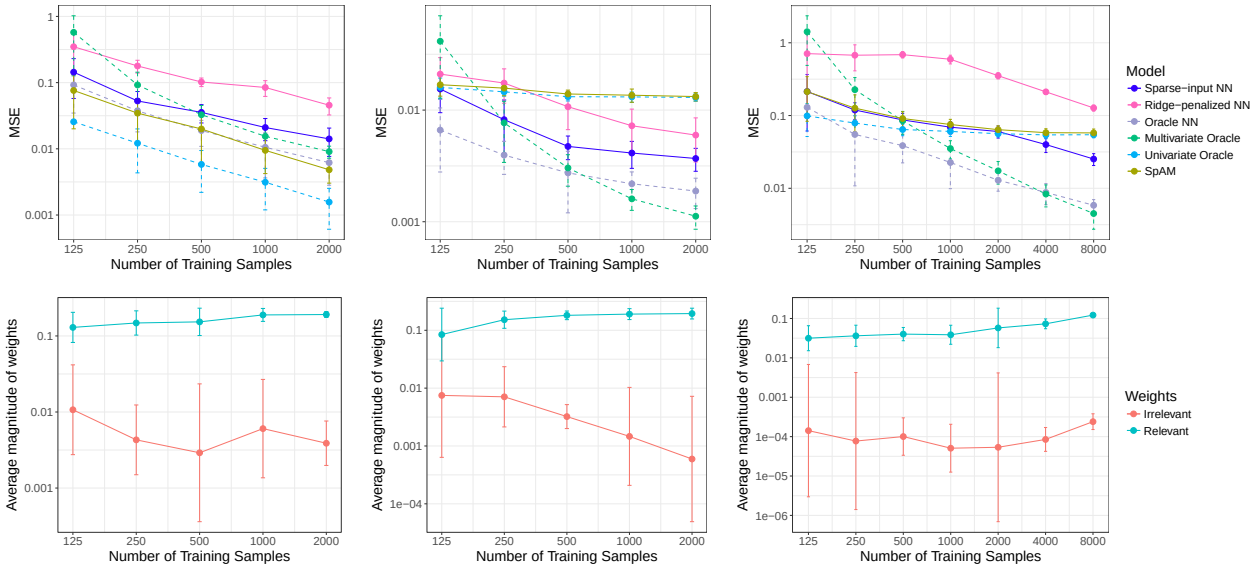
We compare the performance of a sparse-input neural network to a ridge-penalized neural network, SpAM, and a linear model with a lasso penalty. In addition, we fit two special cases of sparse-input neural networks: a lasso-only version and a group-lasso-only version. The penalty parameters in all methods were tuned via 5-fold cross-validation. All the neural networks used three hidden nodes.

Table 1 shows the average performance across 30 random splits of the dataset, as measured by their average MSE on the test set. Sparse-input neural networks had the smallest MSE out of all the methods, and this difference was statistically significant. These results suggest that riboflavin production rates are not well-approximated using an additive univariate or a linear model. Instead it is better to estimate higher-order interactions, even at such small sample sizes. Ridge-penalized neural networks performed the worst, presumably because it was unable to determine which features were relevant for modeling the response. The lasso- and group-lasso-only sparse-input neural networks also did not perform as well. Thus it is valuable to encourage sparsity at the group-level and the level of individual weights.

From this data analysis, we conclude that sparse-input neural networks can be quite effective in practice. Even though neural networks are not typically applied to such high-dimensional datasets, we see here that neural networks with proper regularization can significantly outperform traditional nonparametric regression methods.

7. Discussion

We have introduced using sparse-input neural networks as a nonparametric regression method for high-dimensional data, where the first-layer weights are penalized with a sparse group lasso. When the true model is best approximated by a sparse network, we show that the fitted model using the Lasso has a prediction error that grows logarithmically with the number of features. Thus sparse-



(a) Additive Univariate (b) Complex Multivariate (c) High-Dimensional

Figure 2. Simulation results from the three scenarios: additive univariate function with $p = 50$ (left), a complex multivariate function $p = 50$ (middle), and the sum of multivariate functions with $p = 1000$ (right). The top plots compares the mean squared error (MSE) of nonparametric regression methods. The dashed lines indicate oracle models. The bottom plots show the average magnitude of the weights from the sparse-input neural net fit, stratified into the relevant and irrelevant features.

Table 1. Average performance of the different methods for predicting riboflavin production rates in *Bacillus subtilis*. Standard error given in parentheses.

Method	MSE on test set
Sparse-input NN	**0.1243** (0.010)
Sparse-input NN: lasso-only	0.1799 (0.021)
Sparse-input NN: group-lasso-only	0.1469 (0.016)
Ridge-penalized NN	0.2373 (0.018)
SpAM	0.1445 (0.014)
Linear model with Lasso	0.1448 (0.017)

input neural networks can be effective for modeling high-dimensional data. The proof techniques are quite general and can be extended to other non-smooth penalties. We have also provided empirical evidence via simulation studies and a data analysis to show that sparse-input neural networks can outmatch other more traditional nonparametric regression methods. Our results show that neural networks can indeed be applied to high-dimensional datasets, as long as proper regularization is applied.

One drawback of sparse-input neural networks, and neural networks in general, is that they require a significant amount of time to train. Much of the training time is spent on tuning the hyper-parameters and testing different initializations since the objective function is non-convex. On the other hand, since sparse additive models are convex and have fewer hyper-parameters, they are faster to fit.

There are many directions for future research. Even though this paper only considers sparse-input neural networks with a single hidden layer, the estimation method can be easily extended to multiple hidden layers. It will be important to understand if additional layers are useful in high-dimensional settings and how sparsity-inducing penalties can be best used to control the expressive power of deep networks. To extend our theoretical results, we will need to characterize how the entropy of neural networks increases with the number of layers. In addition, we would like to analyze the behavior of neural networks when the estimator arises from a local minimum. Our theoretical results assume the estimator is located at a global minimum, but this is usually difficult to compute. Also our convergence rates apply when the true function is best approximated by a neural network with sparse first-layer weights. We are working on the case where a sparse function is best approximated by a dense neural network. Finally we plan to extend sparse-input neural networks to classification problems by applying a sigmoid function to the output node.

Acknowledgements

Jean Feng was supported by NIH grants DP5OD019820 and T32CA206089. Noah Simon was supported by NIH grant DP5OD019820.

References

Albertini, Francesca, Sontag, Eduardo D, and Maillot, Vincent. Uniqueness of weights for neural networks. *Artificial Neural Networks for Speech and Vision*, pp. 115–125, 1993.

Alvarez, Jose M and Salzmann, Mathieu. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pp. 2270–2278, 2016.

Anthony, Martin and Bartlett, Peter L. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

Bach, Francis. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017. URL http://jmlr.org/papers/v18/14-546.html.

Barron, Andrew R. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14(1): 115–133, 1994.

Bartlett, Peter L. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44(2):525–536, 1998.

Barzilai, Jonathan and Borwein, Jonathan M. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

Beck, Amir and Teboulle, Marc. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Breiman, Leo, Friedman, Jerome, Stone, Charles J, and Olshen, Richard A. *Classification and regression trees*. CRC press, 1984.

Bühlmann, Peter, Kalisch, Markus, and Meier, Lukas. High-dimensional statistics with a view toward applications in biology. *Annual Review of Statistics and Its Application*, 1:255–278, 2014.

Daubechies, Ingrid, Defrise, Michel, and De Mol, Christine. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11):1413–1457, 2004.

Ghadimi, Saeed and Lan, Guanghui. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1-2):59–99, 2016.

Gong, Pinghua, Zhang, Changshui, Lu, Zhaosong, Huang, Jianhua, and Ye, Jieping. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In *ICML (2)*, pp. 37–45, 2013.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649. IEEE, 2013.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Meier, Lukas, Van de Geer, Sara, Bühlmann, Peter, et al. High-dimensional additive modeling. *The Annals of Statistics*, 37(6B):3779–3821, 2009.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Nadaraya, Elizbar A. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

Nelder, John A and Mead, Roger. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

Nesterov, Yurii. *Introductory lectures on convex optimization: A basic course*, volume 87. Applied Optimization, 2004.

Ravikumar, Pradeep, Liu, Han, Lafferty, John, and Wasserman, Larry. Spam: Sparse additive models. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pp. 1201–1208. Curran Associates Inc., 2007.

Scardapane, Simone, Comminiello, Danilo, Hussain, Amir, and Uncini, Aurelio. Group sparse regularization for deep neural networks. *arXiv preprint arXiv:1607.00485*, 2016.

Simon, Noah, Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Socher, Richard, Perelygin, Alex, Wu, Jean Y, Chuang, Jason, Manning, Christopher D, Ng, Andrew Y, Potts, Christopher, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, pp. 1642, 2013.

Städler, Nicolas, Bühlmann, Peter, and Van De Geer, Sara. 1-penalization for mixture regression models. *Test*, 19 (2):209–256, 2010.

Sun, Xiang. *The Lasso and its implementation for neural networks*. PhD thesis, National Library of Canada= Bibliothèque nationale du Canada, 1999.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

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

Watson, Geoffrey S. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 359–372, 1964.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.