Sparse-Input Neural Networks for High-dimensional Nonparametric Regression and Classification

Jean Feng and Noah Simon *
Department of Biostatistics, University of Washington

June 25, 2019

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 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. This 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 optimal neural network: we show that the excess risk 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 data analyses, we show that these sparse-input neural networks outperform existing nonparametric high-dimensional estimation methods when the data has complex higher-order interactions.

Keywords: Feature selection, Regularization, Interactions, Lasso

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

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 its functional form may be difficult to know apriori. In low and moderate dimensional settings, there are many methods that have been effective for estimating such complex relationships. For classification problems, popular methods include kernel extensions of Support Vector Machines, k-nearest neighbors, and classification trees (Cristianini & Shawe-Taylor 2000, Hofmann et al. 2008); for regression problems, popular methods include spline regression, kernel regression and regression trees Nadaraya (1964), Watson (1964), Breiman et al. (1984). Neural networks have also proven to be particularly effective in problems from 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 example, in genome-wide, and epigenome-wide studies, it is common to collect thousands or millions of features on each of hundreds or thousands of subjects. For general problems in this setting, fully nonparametric methods like random forests or neural networks are rarely used since the number of training observations required for good performance is prohibitive. Instead, it is more typical to apply the Lasso (Tibshirani 1996) or its additive non-parametric 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 using the sum of a small number of univariate (or very low-dimensional) functions. Unfortunately in actual scientific problems, the response may depend on complex interactions between multiple covariates, and failing to model these interactions can result in highly biased estimates.

As existing estimation methods are ineffective for high-dimensional problems with complex interactions, we propose to address this gap using penalized neural networks. In particular, we leverage a unique quality of neural nets that sets them apart from more traditional nonparametric methods: With relatively few parameters, a neural net is able to approximate models with multivariate interaction terms (Barron 1993). This is in contrast to the exponential number of terms necessary required by polynomial or spline regression to model general multivariate functions.

In this paper, we propose controlling the size of the neural network space using the sparse group lasso penalty, which is a mixed ℓ_1/ℓ_2 penalty (Simon et al. 2013). Our method, sparse-input neural networks (SPINN), groups weights connected to the same input node and consequently selects a small subset of informative features for modeling a potentially complex response. We also provide a generalized gradient descent algorithm for training the network.

To understand the theoretical properties of these sparse-input neural networks, we prove oracle inequalities that give probabilistic performance guarantees, assuming we have reached a global minimizer 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, then the difference between the prediction error of a sparse-input neural network and the prediction error of the best neural network shrinks at a rate of $O_p(n^{-1}s^{5/2}\log p)$ (here we treat the number of hidden nodes and

layers 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. In addition, we show that the weights connected to the irrelevant input features also converge to zero. To our knowledge, there have been no theoretical results on the shrinkage rates of irrelevant model parameters for neural networks up to now.

The paper is organized as follows. Section 2 describes our extension of neural networks for high-dimensional problems and an algorithm to train the network. A discussion of work related to our method is provided in Section 3. Section 4 establishes theoretical guarantees of our model. We present simulation studies in Section 5 to better understand how SPINN behaves empirically. Finally, we analyze high- and moderate-dimensional data in Section 6 and find that SPINN can significantly outperform more traditional nonparametric high-dimensional methods when the true function is composed of complex higher-order interaction terms.

2 Sparse-input neural networks

In this paper, we consider neural networks with a single output node and L hidden layers, where hidden layer $a = 1, ..., L$ has m_a hidden nodes. For convenience, define $m_{L+1} = 1$ and $m_0 = p$. The neural network parameters are denoted by $\boldsymbol{\eta} = \{(\boldsymbol{\theta}_a, \boldsymbol{t}_a)\}_{a=1}^{L+1}$ where hidden layer a has weights $\boldsymbol{\theta}_a \in \mathbb{R}^{m_a \times m_{a-1}}$ and intercepts $\boldsymbol{t}_a \in \mathbb{R}^{m_a \times 1}$. The total number of parameters in the neural network is $q = \sum_{a=1}^{L+1} m_a(m_{a-1} + 1)$. In this paper, we suppose that all hidden nodes have the same activation function $\psi : \mathbb{R} \mapsto \mathbb{R}$, which we assume to be bounded and differentiable. For simplicity, $\psi(\boldsymbol{v})$ for a matrix $\boldsymbol{v}$ will denote applying ψ to each element in $\boldsymbol{v}$.

The output of the neural network is generated by propagating values between the neural network layers in the following recursive manner: the hidden node value at layer $a = 1, ..., L$ is defined as

$$\boldsymbol{z}_a(\boldsymbol{x}) = \psi\left(\boldsymbol{\theta}_a \boldsymbol{z}_{a-1}(\boldsymbol{x}) + \boldsymbol{t}_a\right) \tag{1}$$

where $\boldsymbol{z}_0(\boldsymbol{x}) = \boldsymbol{x} \in \mathbb{R}^{m_0 \times 1}$. For regression problems, the final neural network output is

$$f_{\boldsymbol{\eta}}(\boldsymbol{x}) = \boldsymbol{\theta}_{L+1} \boldsymbol{z}_L(\boldsymbol{x}) + t_{L+1}. \tag{2}$$

A neural network for probabilistic binary classification is similar to (2) except that we also apply the sigmoid function $\sigma(z) = 1/(1 + \exp(-z))$ at the end to ensure output values are between zero and one:

$$f_{\boldsymbol{\eta}}(\boldsymbol{x}) = \sigma\left(\boldsymbol{\theta}_{L+1} \boldsymbol{z}_L(\boldsymbol{x}) + t_{L+1}\right). \tag{3}$$

From henceforth in this manuscript, the upper layer network parameters refer to all the parameters above the first hidden layer, i.e. $\boldsymbol{\eta}_{\text{upper}} = \{(\boldsymbol{\theta}_a, \boldsymbol{t}_a)\}_{a=2}^{L+1}$.

By the Universal Approximation Theorem (Leshno et al. 1993, Barron 1993), a neural network with a single hidden layer can approximate any function to any desired accuracy as long as the activation function is not a polynomial and there are a sufficient number of hidden nodes. Therefore neural networks can be thought of as a nonparametric estimator if

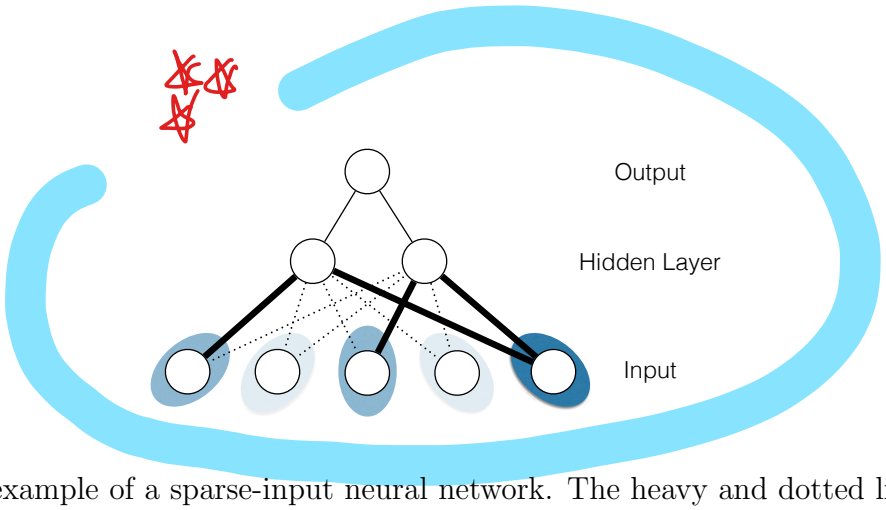


Figure 1: An example of a sparse-input neural network. The heavy and dotted lines indicate nonzero and zero weights, respectively. Each shaded oval corresponds to a group of first-layer weights. The weights from the dark blue oval are both nonzero. Each medium blue oval has a single nonzero weight, so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

we allow the number of hidden nodes/layers to grow as the number of observations increases. We may also think of neural networks nonparametric estimators from the viewpoint of sieve estimation: as we grow the number of hidden nodes in a network with a single hidden layer, the function class is monotonically increasing.

Let $\ell(y, z) : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^+$ denote the loss function, where y represents the observed response and z represents the prediction from the neural network. We suppose that ℓ is differentiable with respect to z. For regression problems, the loss function is typically the squared error loss $\ell(y, z) = (y - z)^2$. For classification problems, the loss function is typically the logistic loss $\ell(y, z) = -y \log z - (1 - y) \log(1 - z)$.

Given observations $(\boldsymbol{x}_i, y_i)$ 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 upper-layer weights using a ridge penalty. Let $\boldsymbol{\theta}_{1,\cdot,j}$ denote the first-layer weights that are multiplied with the jth covariate (the jth column in the matrix $\boldsymbol{\theta}_1$). We fit SPINN by solving

$$\arg\min_{\eta} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f_{\eta}(\boldsymbol{x}_i)\right) + \lambda_0 \sum_{a=2}^{L+1} \|\boldsymbol{\theta}_a\|_2^2 + \lambda \sum_{j=1}^{p} \Omega_{\alpha}(\boldsymbol{\theta}_{1,\cdot,j}) \tag{4}$$

where $\alpha \in [0, 1]$ and $\Omega_{\alpha}(\boldsymbol{\theta}) = (1 - \alpha)\|\boldsymbol{\theta}\|_1 + \alpha\|\boldsymbol{\theta}\|_2$ is the sparse group lasso. Thus α can be thought of as a balancing parameter between the Lasso and the group lasso (Yuan & Lin 2006): when $\alpha = 0$, the sparse group lasso reduces to the former and when $\alpha = 1$, it reduces to the latter.

We have three types of penalties in this criterion. The ridge penalty $\|\cdot\|_2^2$ serves to control the magnitude of the weights that are not in the first layer. The sparse group lasso Ω_{α} is a mixture of the group lasso and lasso penalties (Simon et al. 2013). The group lasso penalty on $\boldsymbol{\theta}_{1,\cdot,j}$ encourages sparsity at the input level by encouraging the entire vector $\boldsymbol{\theta}_{1,\cdot,j}$ to be zero. The lasso penalty on $\boldsymbol{\theta}_1$ encourages sparsity across all the weights, so the hidden nodes are encouraged to connect to only a few input nodes. The parameter α allows us to control the level of sparsity at the level of the inputs vs. the individual weights. Pictorially, (4) encourages fitting neural networks like that in Figure 1.

One could also consider adding a sparsity penalty to upper layer parameters. This is useful if the upper hidden layers have too many nodes and many need to be pruned away.

However in the high-dimensional setting, performance is usually better for networks that have a small number of hidden layers and the upper hidden layers have a small number of nodes.

2.1 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 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 (4). This algorithm is similar to that in Alvarez & Salzmann (2016).

Let $\mathcal{L}_{x,y}^{\text{smooth}}(\boldsymbol{\eta})$ be the smooth component of the loss function in (4) defined as follows

$$\mathcal{L}_{x,y}^{\text{smooth}}(\boldsymbol{\eta}) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, f_{\boldsymbol{\eta}}(\boldsymbol{x}_i)\right) + \lambda_0 \sum_{a=2}^{L+1} \|\boldsymbol{\theta}_a\|_2^2. \tag{5}$$

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

$$\left(S\left(\boldsymbol{z}, c\right)\right)_j = \text{sign}(z_j)\left(|z_j| - c\right)_+ . \tag{6}$$

The algorithm for training a sparse-input neural network is given in Algorithm 1. The proximal gradient step is composed of three sub-steps. The first sub-step (8) 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 sub-steps, (9) and (10), are the proximal operations for the Sparse Group Lasso (Simon et al. 2013): a soft-thresholding operation followed by a soft-scaling operation on each $\boldsymbol{\theta}_{1,\cdot,j}$. Therefore training a sparse-input neural networks should take a similar amount of time as training a traditional neural network where the objective function only contains a ridge penalty.

At each iteration of the algorithm, the step size γ_k is initialized with some value in $[\gamma_{\min}, \gamma_{\max}]$ and then tuned according to a monotone line search criterion. In our implementation, we simply used a fixed value, e.g. $\gamma_{min} = \gamma_{max}$, though one can also adaptively choose the initial step size (Barzilai & Borwein 1988, Gong et al. 2013). Let $\boldsymbol{\eta}^{(k-1,2)}$ be the model parameters at the $k-1$-th iteration and $\boldsymbol{\eta}^{(k,2)}$ be the proposed model parameters for step size γ_k at iteration k. The monotone line search criterion accepts step size γ_k if the following condition is satisfied:

$$L(\boldsymbol{\eta}^{(k,2)}) \leq L(\boldsymbol{\eta}^{(k-1,2)}) - t\gamma_k \|\boldsymbol{\eta}^{(k,2)} - \boldsymbol{\eta}^{(k-1,2)}\|^2 \tag{7}$$

where $L(\cdot)$ is the objective function of (4) and $t \in (0,1)$. This line search criterion guarantees that Algorithm 1 converges 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.

Algorithm 1 Training sparse-input neural networks

Initialize neural network parameters $\boldsymbol{\eta}^{(0,2)}$. Choose $s \in (0,1)$ and $\gamma_{min}, \gamma_{max}$ such that $\gamma_{max} \geq \gamma_{min} > 0$.

for iteration $k = 1, 2, ...$ **do**

$\gamma_k \in [\gamma_{min}, \gamma_{max}]$

repeat

Proximal_operator

$$\boldsymbol{\eta}^{(k,0)} = \boldsymbol{\eta}^{(k-1,2)} - \gamma_k \nabla_\eta \mathcal{L}_{x,y}^{\text{smooth}}(\boldsymbol{\eta}^{(k-1,2)}) \tag{8}$$

$$\boldsymbol{\theta}_1^{(k,1)} = S\left(\boldsymbol{\theta}_1^{(k,0)}, \gamma_k \lambda (1-\alpha)\right) \tag{9}$$

↳ LASSO_operator

for $j = 1, ..., p$ **do**

$$\boldsymbol{\theta}_{1,\cdot,j}^{(k,2)} = \left(1 - \frac{\gamma_k \lambda \alpha}{\left\|\boldsymbol{\theta}_{1,\cdot,j}^{(k,1)}\right\|_2}\right)_+ \boldsymbol{\theta}_{1,\cdot,j}^{(k,1)} \tag{10}$$

↳ group-LASSO_operator.

end for

for $a = 2, ..., L+1$ **do**

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

↳ 나머지-Layer

end for

$\gamma_k := s\gamma_k$

until line search criterion is satisfied

end for

2.2 Tuning Hyper-parameters

There are two types of hyper-parameters for fitting a sparse-input neural network: the penalty parameters for the ridge and sparse group lasso penalties and those specifying the number of layers and number of nodes per layer in the network. The hyper-parameters should be tuned to ensure low generalization error of the model and are typically chosen via cross-validation.

When fitting sparse-input neural networks for high- or moderate-dimensional problems, we found that the hyper-parameter values can be tuned using either grid search or gradient-free optimization methods such as Nelder-Mead (Nelder & Mead 1965) or Bayesian optimization (Snoek et al. 2012). The former method is typically used when there are three or fewer hyper-parameters and the latter class is typically used when there are many hyper-parameters. In our implementation, we typically consider a number of possible neural network structures and different penalty parameter values; grouping the hyper-parameters in this manner result in four hyper-parameters to tune in SPINN. Thus the number of hyper-parameters is only slightly more than the typical use case in grid search.

The computation time for tuning hyper-parameters can be further reduced by pretuning the range of the possible hyper-parameter values. In particular, the optimal network structure for such high-dimensional problems tends to be small since large networks easily overfit to the data. For the examples in this paper, we only consider network structures with no more than three hidden layers and no more than fifty hidden nodes per layer. In addition, since the upper layers of the network are parameterized by a small number of parameters, we found that the generalization error is relatively insensitive to the ridge penalty parameter. For all of our empirical analyses, we pretune the ridge penalty parameter and fix it to a small value (usually ≤ 0.001).

The rest of the hyper-parameters play a more influential role on the generalization error. Understanding their affects on the performance not only increases our understanding of SPINN but also provides guidance for tuning their values. We analyze the role of the network structure and the sparse group lasso penalty parameter λ from a theoretical standpoint in Section 4 and present an empirical analysis of the role of the balancing parameter α in Section 5.

3 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 — this is in the low dimensional setting with $n \gg p$. The recent work by Bach (2017) is most closely aligned to our work: he considers convex neural networks which have a single hidden layer with 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 using a ridge penalty on the internal network architecture was not necessary for a neural network to have good performance; instead, using an appropriate network architecture led to larger decreases in generalization error. Our results support this claim since the sparse group lasso also performs structure-learning.

Previously, statistical convergence rates for neural networks have been established for problems when the ℓ_1-norm of the weights are constrained. These results show that the estimation error of the neural networks indeed grew with the logarithm of the number of input nodes p (Bartlett 1998, Anthony & Bartlett 2009). Our convergence rate results also have a $\log p$ term. However, we improve upon previously established convergence rates by showing that the excess estimation error shrinks at a rate of n^{-1} rather than $n^{-1/2}$ as was given in previous bounds. This faster convergence rate allows us to additionally bound the rate at which the norm of the irrelevant network weighs shrink to zero.

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.

4 Theoretical Guarantees

In this section, we provide probabilistic, finite-sample upper bounds on the prediction error of sparse-input neural networks. Instead of using a ridge penalty, we constrain the norm of the upper-layer weights. Our sparse-input neural network problem now becomes

$$\hat{\boldsymbol{\eta}} \in \arg\min_{\eta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f_\eta(\boldsymbol{x}_i)\right) + \lambda \sum_{j=1}^{p} \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j}) \tag{11}$$
$$\text{where } \Theta = \{\boldsymbol{\eta} \in \mathbb{R}^q : \|\text{other}(\boldsymbol{\eta})\|_2 \leq K\}$$

for a constant $K > 0$ and $\text{other}(\boldsymbol{\eta}) = (\boldsymbol{t}_1, \{(\boldsymbol{\theta}_a, \boldsymbol{t}_a)\}_{a=2}^{L+1})$. All of our proofs are in the Supplementary Materials.

Notice that (11) 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

Suppose covariates X have support $\mathcal{X} \subseteq [-X_{\max}, X_{\max}]^p$ and $\mathcal{X}$ contains some open set. In addition, suppose $Y = f^*(X) + \epsilon$ where ϵ is a random variable with mean zero. Let $\mathbb{P}$ denote the expectation with respect to the joint distribution of the covariates X and response Y. Given n observations, we denote the empirical distribution as $\mathbb{P}_n$.

The neural network can be defined either as in (2) or (3), where we suppose the activation function ψ is the hyperbolic tangent function. Note that neural networks with the sigmoid function are included in this framework since tanh and the sigmoid function are related by a linear transformation.

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

$$EQ(\boldsymbol{\eta}) = \left\{ \boldsymbol{\eta}' \in \Theta : f_{\boldsymbol{\eta}'}(\boldsymbol{x}) = f_{\boldsymbol{\eta}}(\boldsymbol{x}) \forall \boldsymbol{x} \in \mathcal{X} \right\}. \tag{12}$$

If $\boldsymbol{\eta}$ is "generic" as defined in Fefferman (1994), $EQ(\boldsymbol{\eta})$ only contains parameterizations that are sign-flips or permutations of $\boldsymbol{\eta}$ and thus $EQ(\boldsymbol{\eta})$ has cardinality of at most $\prod_{a=1}^{L} 2^{m_a}(m_a!)$. Roughly, a generic NN parameter is one where all hidden nodes have nonzero input weights, $\boldsymbol{\theta}_{a,j,\cdot} \neq \mathbf{0}$, and no two hidden nodes are sign-flips of each other, i.e. $|\boldsymbol{\theta}_{a,j',\cdot}| \neq |\boldsymbol{\theta}_{a,j,\cdot}|$ for $j \neq j'$.

Let the set of optimal neural networks that minimize the expected loss be denoted

$$EQ^* = \arg\min_{\boldsymbol{\eta} \in \Theta} \mathbb{P}\ell(y, f_{\boldsymbol{\eta}}(\boldsymbol{x})). \tag{13}$$

We suppose that EQ^* is the union of $Q \geq 1$ equivalence classes, where all of the optimal neural networks are generic. We will suppose that Θ is chosen large enough such that the gradient of the expected loss at every element in EQ^* is zero.

Next suppose that the expected loss is minimized using neural networks that employ only a few input features S, which we call the "relevant" features. That is, for $\boldsymbol{\eta}^* \in EQ^*$, we suppose that all weights tied to the irrelevant features S^c are zero. Note that all neural nets from the same equivalence class have the same set of features with nonzero weights because members of the equivalence class are permutations and/or sign-flips of one another. In this work we are particularly interested in the case where the optimal neural networks are sparse, e.g. those where the cardinality of S, denoted $|S|$, is small. For any $\boldsymbol{\eta} \in \Theta$, let $\boldsymbol{\theta}_{1,\cdot,S}$ denote the weights tied to the input nodes S and $\boldsymbol{\theta}_{1,\cdot,S^c}$ denote the weights tied to the input nodes S^c. Let $\boldsymbol{\eta}_S$ denote the NN parameters that are exactly the same as $\boldsymbol{\eta}$ except where $\boldsymbol{\theta}_{1,\cdot,S^c}$ is set to zero.

For any $\boldsymbol{\eta}$, let the closest element in EQ^* be defined as

$$\boldsymbol{\eta}^{*(\boldsymbol{\eta})} = \{(\boldsymbol{\theta}_a^{*(\boldsymbol{\eta})}, \boldsymbol{t}_a^{*(\boldsymbol{\eta})})\}_{a=1}^{L+1} \in \arg\min_{\boldsymbol{\eta}^* \in EQ^*} \|\boldsymbol{\eta} - \boldsymbol{\eta}^*\|_2,$$

where we randomly pick one optimal network if there are multiple that minimize the distance to $\boldsymbol{\eta}$. Define the excess loss of a neural network with parameters $\boldsymbol{\eta}$ as

$$\mathcal{E}(\boldsymbol{\eta}) = \mathbb{P}\ell(y, f_{\boldsymbol{\eta}}(\boldsymbol{x})) - \min_{\boldsymbol{\eta}' \in \Theta} \mathbb{P}\ell(y, f_{\boldsymbol{\eta}'}(\boldsymbol{x})) \tag{14}$$

$$= \mathbb{P}\left[\ell(y, f_{\boldsymbol{\eta}}(\boldsymbol{x})) - \ell(y, f_{\boldsymbol{\eta}^{*(\boldsymbol{\eta})}}(\boldsymbol{x}))\right]. \tag{15}$$

4.2 Results

To understand the behavior of our estimated neural network from (11), we upper bound the excess loss as well as the norm of the weights connected to the irrelevant inputs. Our proof

technique is inspired by Städler et al. (2010), however significant adaptations were needed to deal with the complex loss surface and equivalence classes of neural networks.

In order for our results to hold, we make the assumption that the expected loss is locally strongly convex at all $\boldsymbol{\eta}^* \in EQ^*$. Since this only specifies the local behavior at EQ^*, this assumption is relatively weak. We use the notation $A \succeq B$ to indicate that $A - B$ is a positive semi-definite matrix. More formally this assumption states:

Condition 1. *Let the neural network parameter $\boldsymbol{\eta}$ be ordered such that*

$$\boldsymbol{\eta} = (\boldsymbol{\theta}_{1,\cdot,S^c}, \boldsymbol{\theta}_{1,\cdot,S}, \boldsymbol{t}_1, \boldsymbol{\theta}_2, \boldsymbol{t}_2, \cdots, \boldsymbol{\theta}_{L+1}, \boldsymbol{t}_{L+1}).$$

There is a constant $h_{min} > 0$ that may depend on $m_1, ..., m_{L+1}, |S|, f^*$ and the distribution $\mathbb{P}$, but does not depend on p, such that for all $\boldsymbol{\eta}^* \in EQ^*$,

$$\left[\nabla^2_{\eta} \mathbb{P}\ell(y, f_{\boldsymbol{\eta}}(\boldsymbol{x}))\right]_{\eta=\eta^*} \succeq h_{min} \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{16}$$

where the top left zero matrix is $m_1|S^c| \times m_1|S^c|$ and the bottom right matrix is the identity matrix of dimension $(q - m_1|S^c|) \times (q - m_1|S^c|)$.

In addition, we need the following identifiability condition.

Condition 2. *For all $\epsilon > 0$, there is an $\chi_\epsilon > 0$ that may depend on $m_1, ..., m_{L+1}, |S|, f^*$ and the distribution $\mathbb{P}$, but does not depend on p, such that*

$$\chi_\epsilon \leq \inf_{\boldsymbol{\eta} \in \Theta} \Big\{ \mathcal{E}(\boldsymbol{\eta}) : \|\boldsymbol{\eta}_S - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}\|_2 \geq \epsilon \text{ and } \|\boldsymbol{\theta}_{1,\cdot,S^c}\|_1 \leq 3 \sum_{j \in S} \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j} - \boldsymbol{\theta}^{*(\boldsymbol{\eta})}_{1,\cdot,j}) + \|\operatorname{other}(\boldsymbol{\eta}) - \operatorname{other}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})})\|_2 \Big\}.$$

Condition 2 places a lower bound on the excess loss of neural networks outside the set of optimal neural networks EQ^*. 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 the other parameters in $\boldsymbol{\eta}$ and $\boldsymbol{\eta}^{*(\boldsymbol{\eta})}$. By restricting to this smaller set of neural networks, it is more realistic to claim that χ_ϵ is independent of p. This condition is similar to compatibility conditions used in proving theoretical results for the Lasso (Bühlmann & Van De Geer 2011).

Finally, we require a bound on the third derivative of the expected loss. Since $\mathcal{X}$ and the upper layer weights are bounded, It is easy to show that this condition is satisfied when the loss function ℓ is mean squared error or logistic loss.

Condition 3. *The third derivative of the expected loss function is bounded uniformly over Θ by some constant $G > 0$ that may depend on $m_1, ..., m_{L+1}, |S|, f^*, K$ and the distribution $\mathbb{P}$, but does not depend on p:*

$$\sup_{\boldsymbol{\eta} \in \Theta} \max_{j_1 j_2 j_3} \left| \frac{\partial^3}{\partial \eta_{j_1} \partial \eta_{j_2} \partial \eta_{j_3}} \mathbb{P}\ell(y, f_{\boldsymbol{\eta}}(\boldsymbol{x})) \right| \leq G. \tag{17}$$

With the three conditions above, 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*

$$\mathcal{T}_{\tilde{\lambda},T} = \Bigg\{ \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n : \sup_{\boldsymbol{\eta} \in \Theta} \frac{\left|(\mathbb{P}_n - \mathbb{P})\left(\ell(y, f_{\eta^*(\boldsymbol{\eta})}(\boldsymbol{x})) - \ell(y, f_{\boldsymbol{\eta}}(\boldsymbol{x}))\right)\right|}{\tilde{\lambda} \vee \left(\|\text{other}(\boldsymbol{\eta}) - \text{other}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})})\|_2 + \sum_{j=1}^p \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j} - \boldsymbol{\theta}^{*(\boldsymbol{\eta})}_{1,\cdot,j})\right)} \leq T\tilde{\lambda} \Bigg\}.$$

Suppose Conditions 1, 2, and 3 hold. Suppose that the optimal neural networks EQ^* only have nonzero weights for input features in S. Let $\hat{\boldsymbol{\eta}}$ be a solution of (11). Then over the set $\mathcal{T}_{\tilde{\lambda},T}$, we have for any $\lambda \geq 2\tilde{\lambda}T$

$$\mathcal{E}(\hat{\boldsymbol{\eta}}) + 2\left(\lambda - T\tilde{\lambda}\right) \sum_{j \in S^c} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j}) \leq \left(T\tilde{\lambda} + \lambda\right)^2 ((1-\alpha)\sqrt{m_1} + \alpha)^2 |S| C_0^2 \tag{18}$$

where

$$C_0^2 = \frac{1}{\epsilon_0} \vee \frac{C_1^2 (K + \max_{\eta^* \in EQ^*} \sum_{j \in S} \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j^*}))^2}{\chi_{\epsilon_0}}, \tag{19}$$

$$\epsilon_0 = \frac{h_{min}(1 - \alpha + \alpha/\sqrt{m_1})^3}{C_1 G\left((1-\alpha)\sqrt{m_1|S|} + \alpha\sqrt{|S|} + \sqrt{m^*}\right)^3} \tag{20}$$

for some constant $C_1 > 0$.

Theorem 1 simultaneously bounds the excess loss and the norm of the irrelevant nodes. Consider the case where the identifiability constant χ_{ϵ_0} is sufficiently large such that $C_0^2 = \epsilon_0^{-1}$. Then the above theorem states that for $\alpha \in (0, 1)$ (e.g. we are using the sparse group lasso and not the degenerate cases with only the lasso or only the group lasso), the excess loss will be on the order of $O_p(\tilde{\lambda}^2 G m_1^{5/2} |S|^{5/2})$ and the norm of $\hat{\boldsymbol{\theta}}_{S^c}$ will shrink at the rate of $O_p(\tilde{\lambda} G m_1^{5/2} |S|^{5/2})$. 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). If $\tilde{\lambda}$ shrinks as the number of samples increases, these values will go to zero. The rate of convergence is fastest if we set $\tilde{\lambda}$ to a very small value; however we must choose $\tilde{\lambda}$ carefully so that $\mathcal{T}_{\tilde{\lambda},T}$ occurs with high probability.

Our convergence rate of here depends on the number of hidden nodes in the first layer and implicitly depends on the upper layers contribute to the rate through the constant G in Condition 3. If we want the overall estimation error, we must also take into account how the network structure contributes via the approximation error. Increasing the number of nodes and layers in the network will decrease the approximation error; however the variance of model will increase and our upper-bound in (18) will be large.

The relevance of Theorem 1 depends on the probability of the set $\mathcal{T}_{\tilde{\lambda},T}$. To analyze this sets probability, we introduce one more condition that bounds the gradient of the loss with respect to the nodes in the *first hidden layer* and *the network parameters in the upper layers*. Similar to Condition 3, it is easy to show that this condition is satisfied when the loss function is mean squared error or logistic loss.

Condition 4. *Define the upper portion of the neural network f_η as $f_{\eta_{\text{upper}}} : \mathbb{R}^{m_1} \mapsto \mathbb{R}$. The gradient of the loss of $f_{\boldsymbol{\eta}_{\text{upper}}}$ is uniformly bounded by:*

$$\sup_{\boldsymbol{\eta}\in\Theta,\boldsymbol{x}\in\mathcal{X}} \left\| \nabla_{\boldsymbol{z},\boldsymbol{\eta}_{\text{upper}}} \ell(y, f_{\boldsymbol{\eta}_{\text{upper}}}(\boldsymbol{z}))\Big|_{\boldsymbol{z}=\psi(\boldsymbol{\theta}_1^\top \boldsymbol{x}+\boldsymbol{t})} \right\|_\infty \leq M_1(|y| + M_0) \tag{21}$$

where constants $M_0, M_1 > 0$ may depend on $m_1, ..., m_{L+1}, |S|, f^*, K$, and $X_{\max}$, but does not depend on p.

We specialize the following theorem to classification and regression settings. The proof relies on techniques from empirical process theory. Let m_{upper} be the number of network parameters in the upper layers of the network (i.e. exclude the weights in the first layer).

Theorem 2. *Consider the following two settings:*

1. *Regression setting: Let*
$$y = f^*(\boldsymbol{x}) + \epsilon$$
where ϵ is a sub-gaussian random variable with mean zero. That is, ϵ satisfies for some constants τ and σ_0,
$$\tau^2(\mathbb{E}e^{|\epsilon|^2/K_\epsilon^2} - 1) \leq \sigma_0^2. \tag{22}$$
We suppose that ϵ is independent of $\boldsymbol{x}$. Suppose $\ell(\cdot,\cdot)$ is squared-error loss.

2. *Classification setting: Let Y take on values $\{0, 1\}$ where $p(Y = 1|\boldsymbol{x}) = f^*(\boldsymbol{x})$. Suppose $\ell(\cdot,\cdot)$ is logistic loss.*

Suppose that the set of optimal neural networks EQ^* is composed of Q equivalence classes. For each setting, there exists a constant $c_0 > 0$ that only depends on τ and σ_0 such that for

$$\tilde{\lambda} = c_0 M_1(m_1 + K\sqrt{m_{\text{other}}})\sqrt{\frac{\log n}{n}}\left(\sqrt{\log Q} + \frac{X_{\max}}{c_1}\log(nc_2)\sqrt{\log(c_2 p)}\right). \tag{23}$$

we have for any $T \geq 1$,

$$Pr_{X,Y}\left(\mathcal{T}_{\tilde{\lambda},T}\right) \geq 1 - O\left(\frac{1}{n}\right) \tag{24}$$

where $c_1 = 1 - \alpha + \alpha/\sqrt{m_1}$ and $c_2 = m_1 + K\sqrt{m_{\text{other}}} + X_{\max}/c_1$.

We now combine Theorems 1 and 2 to obtain a final bound on the excess loss and norm of the irrelevant weights. Here we focus primarily on the role of p rather than the role of the number of layers and hidden nodes since in high-dimensional settings, we find shallow networks typically have better performance. In the above results, the effect of the number of layers and hidden nodes in the upper layers are encapsulated in the constants G and M_1, which are used to bound the first and third derivatives of the loss over the entire space of Θ. (It is easy to upper-bound these constants with a rate that grows exponentially in the number of hidden layers, which is similar to previous results (Anthony & Bartlett 2009).)

Thus to combine results, define $M_2 = GM_1(\sqrt{m_{\text{other}}} + m_1)$, which depends on the number of hidden nodes at the hidden layers, $|S|$, f^*, K and $X_{\max}$ but not p. If the identifiability constant is not too small, Theorems 1 and 2 state that if $\alpha \in (0, 1)$ and $\tilde{\lambda}$ is chosen according to (23), the excess loss converges at the rate

$$O_p\left(n^{-1} M_2^2 m_1^{5/2} |S|^{5/2} \log p\right)$$

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

$$O_p\left(n^{-1/2} M_2 m_1^{5/2} |S|^{5/2} \sqrt{\log p}\right)$$

modulo log terms that do not depend on p. Thus we find that the total number of features p only enters these rates in a log-term.

To the best of our knowledge, our results are the first to bound the convergence rate of the norm of irrelevant weights. Though our bounds might not be tight, 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 analyses.

We hope to improve these convergence rate bounds in future research. In particular, we would like to shrink the exponent on m_1 and $|S|$ since these rates become very slow for moderate values of m_1 and $|S|$. However we find that in practice, the number of effective nodes in the first hidden layer is controlled partly by the shrinkage behavior of the lasso since the lasso encourages sparsity both in the inputs as well as the first hidden layer. Thus the convergence rates are likely faster when the true function f^* is closely approximated by a network with a small number of hidden nodes in the first layer. Also, our bound depends on the gradient of the loss of the networks in the model class. It seems possible to tighten the bounds if we can show that the gradient of the fitted network is similar to that of the optimal neural network. Intuitively, if the optimal neural network is smooth, i.e. the magnitude of these derivatives are small, then the fitted network by optimizing the penalized log likelihood is also likely to be smooth.

5 Simulation study

In this section, we present simulations to understand various facets of these sparse-input neural networks. First, we compare empirical convergence rates to our theoretical convergence rates to understand if our rates hold and how tight they are. Second, we would like to understand the roles of the lasso vs. group lasso penalties and how they interact with each other. Finally, we compare SPINN to other methods in various settings.

For all of the simulations, we generated the data according to the model $y = f^*(\boldsymbol{x}) + \sigma\epsilon$ where $\epsilon \sim N(0, 1)$ and σ is scaled such that the signal to noise ratio is 2. We sampled each covariate $x_j \in \mathbb{R}$ independently from the standard uniform distribution. For the simulations, the true function f^* will only depend on the first $|S|$ features, though the number of covariates p may be much larger. Throughout the simulations, the mean squared error is defined as $E[(f^*(x) - f_{\hat{\eta}}(x))^2]$ since the true model is known.

5.1 Confirming the convergence rates

Here we aim to understand if our convergence rates hold in practice. Recall that our theoretical results depend on a number of assumptions. We assumed that the fitting procedure finds the global minima, even though this is difficult to show in practice since our optimization problem is not convex. In addition, Conditions 1 and 2 assumed that the behavior of the expected loss depended on constants $h_{\min}$ and α_ϵ that did not depend on p.

As our theoretical results only bound the excess loss, we would like an experimental setup that is not affected by the approximation error. Here we set the true function to be a neural network with one hidden layer and four hidden nodes:

$$f^*(\boldsymbol{x}) = \tanh(x_1 + 2x_2 - 3x_3 + 2x_4) + 2\tanh(x_1 - 2x_5 + 2x_6) + \tanh(-x_2 - x_3 - x_6) + \tanh(x_5 - 0.5x_3 + 0.5x_6).$$

Thus the excess loss is $E[(f^*(x) - f_{\hat{\eta}}(x))^2]$.

First we check the behavior of SPINN as we grow the number of observations. Here the fitted neural networks have the same structure as the optimal neural network and we keep the parameter α fixed. We estimate the convergence rate of the excess loss by performing linear regression between $\log(\log n/n)$ and the logarithm of the excess loss. Our oracle inequality suggests that the estimated coefficient should be 1. Indeed, we estimate the slope to be 1.03 (Figure 2 a, left). Similarly, we can check the convergence rate of the sparse group lasso penalty of the irrelevant weights. Plotting the sparse group lasso penalty of the irrelevant weights against $\sqrt{\log n/n}$, we see that a near-linear relationship only begins when there are at least 400 observations (Figure 2 a, right). Therefore we fit a linear model between $\log(\log n/n)$ and the logarithm of the weights of the irrelevant weights for the simulations with at least 400 observations. Our oracle inequality suggests that the estimated coefficient should be 0.5 and we estimate that the slope is 0.58, which is slightly faster than our theoretical bound.

Next we check that convergence rate with respect to the number of irrelevant inputs. As before, the fitted networks have the same structure are the optimal network. Here the number of training observations remains constant ($n = 200$). To understand how the excess loss grows with the number of features, we perform linear regression of the excess loss against the number of covariates p and its logarithm. According to our theoretical results, the excess loss should grow with the logarithm of p, not p itself. We estimate a coefficient of 0.0003 for p and a coefficient of 0.031 for $\log p$ (Figure 2 b, left). Similarly, we check how the weight of the irrelevant weights grow with the number of features, we regress the weight against the number of covariates p and the square root of the logarithm of p. We estimate a coefficient of -0.0007 for p and a coefficient of 0.190 for $\sqrt{\log p}$ (Figure 2 b, right). From these results, we conclude that our convergence rates with respect to the number of features and the number of observations appears tight.

Finally, we would like to understand how the number of hidden nodes affects estimation error and discrimination of the relevant vs. irrelevant nodes. Recall that our derived rates depend on large powers of m_1, the number of hidden nodes in the first layer; however we suspect that our upper bounds are not tight with respect to m_1. We keep the number of observations and features constant ($n = 200, p = 50$), but grow the number of nodes in the single hidden layer. Surprisingly, we find that the excess loss is relatively constant as the number of hidden nodes grows (Figure 2 c, left). Likewise, we find that the weight of the

irrelevant nodes does not grow with the number of hidden nodes (Figure 2 c, right). The most likely explanation is that the lasso encourages sparsity in the number of hidden nodes in the first hidden layer since it encourages weights to be set to zero. In the empirical results in the following sections, we find that SPINN can often zero out a large proportion of hidden nodes.

5.2 The role of lasso vs. group lasso

The α parameter in the sparse group lasso balances the shrinkage behavior of the lasso and the group lasso. To understand the effects of the lasso and group lasso on the behavior of SPINN, we performed a simulation study where we varied their penalty parameters (Figure 2d). Here the true function is

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

We generate 50 covariates (where only the first six are relevant), and 250 observations.

There is a near symmetry in using the lasso and the group lasso penalties. The mean squared error (MSE) is small when either the lasso is large and the group lasso is small or the lasso is small and the group lasso is large. For both scenarios, the weights tied to the relevant inputs are large and those tied to the irrelevant ones are small. The MSE is large when the lasso and the group lasso are both large or are both small; this is unsurprising since the former means that we are over-penalizing and the latter means that we are under-penalizing.

Nonetheless there is still a difference between the two penalties: upweighting the lasso leads to a fitted model with a larger proportion of the weights on the irrelevant nodes and a smaller proportion of the weights on the relevant nodes. Since the group lasso penalty is better able to discriminate between inputs, upweighting the group lasso results in models with slightly smaller mean squared error (MSE = 0.009 vs. 0.010).

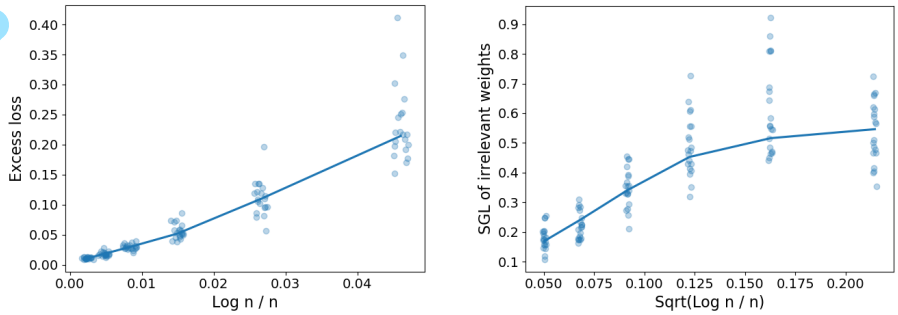
5.3 Comparing against other methods

We now present a simulation study to understand how sparse-input neural networks compare against other methods. We consider 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 cases, the true function is sparse and only depends on the first six variables. We compare sparse-input neural networks to the following methods:

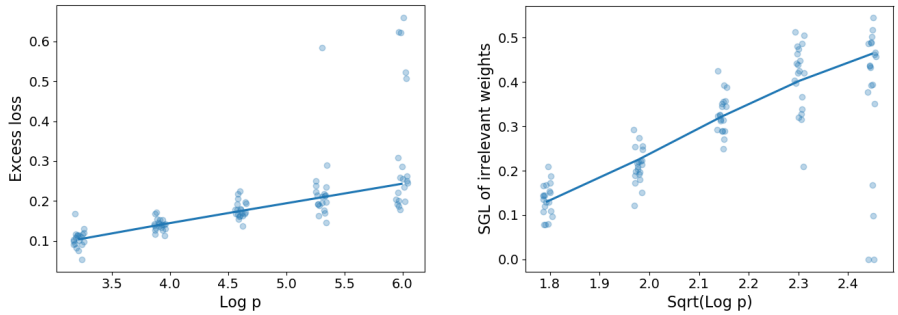
- neural network with a single hidden layer and a ridge penalty on all the weights (ridge-only neural network);
- Sparse Additive Model (SpAM), which fits an additive univariate model with a sparsity-inducing penalty (Ravikumar et al. 2007).
- random forests (Breiman 2001)
- oracle ridge-only neural network involving only the first six covariates;
- oracle additive univariate model of the form $\sum_{i=1}^{6} g_i(x_i)$ where g_i are fit using additive smoothing splines;

Figure 2: Simulation results investigating convergence rates and effects of penalty parameters

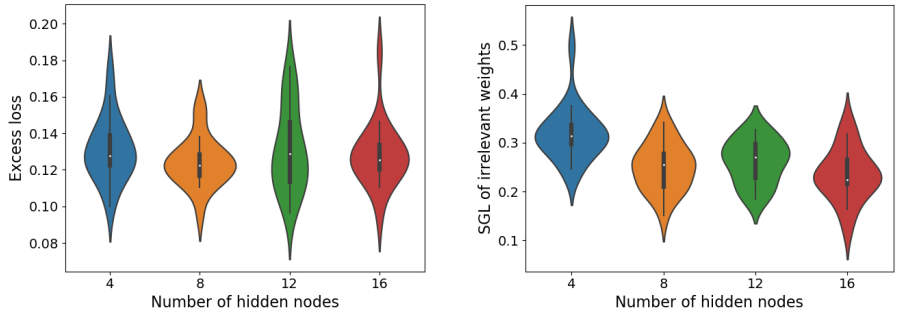
(a) Effects of varying number of observations ($n = 100, 200, ..., 3200$).



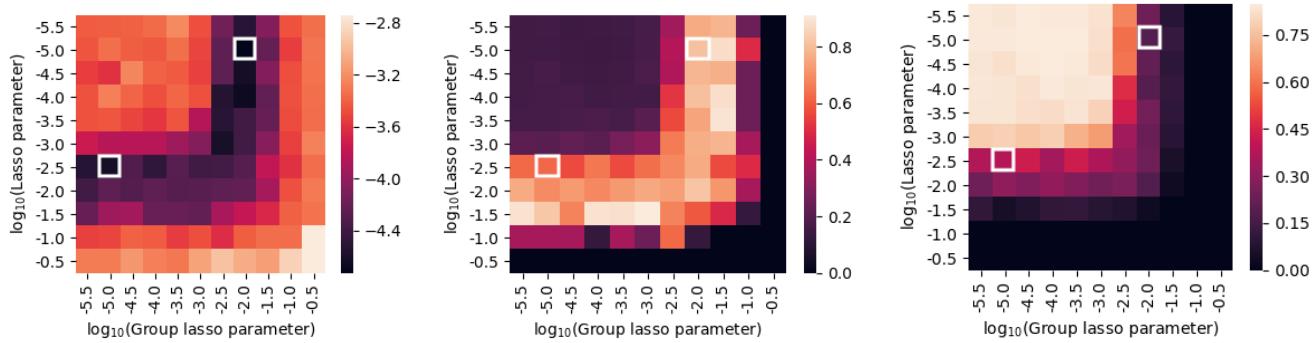
(b) Effects of varying number of covariates ($p = 25, 50, ..., 400$).



(c) Effects of varying number of hidden nodes ($m_1 = 4, 8, 12, 16$).



(d) Effects of varying the lasso and group lasso penalty parameters. The white box in the right and left of each plot corresponds to the penalty parameter setting that has the lowest mean squared error when upweighting the group lasso and the lasso, respectively. The heatmaps plot the mean squared error (left), the proportion weight assigned to relevant nodes (middle), and the proportion weight assigned to the irrelevant nodes (right).



- oracle general multivariate model $g(x_1, ..., x_6)$ where g is fit using a 6-variate smoothing spline.

The last three methods are oracles since they take only the relevant covariates as input. These 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 is assessed by the mean squared error, evaluated over 2000 points drawn from $\mathbb{P}_X$.

We generated a training set of varying sizes and a test set of 2000 observations. The penalty parameters in all the methods were tuned using 3-fold cross validation. When tuning the hyper-parameters in SPINN, we considered up to three hidden layers and up to fifteen hidden nodes per layer. We use tanh as the activation function in our neural networks. Each simulation was repeated 20 times.

5.3.1 Additive univariate model

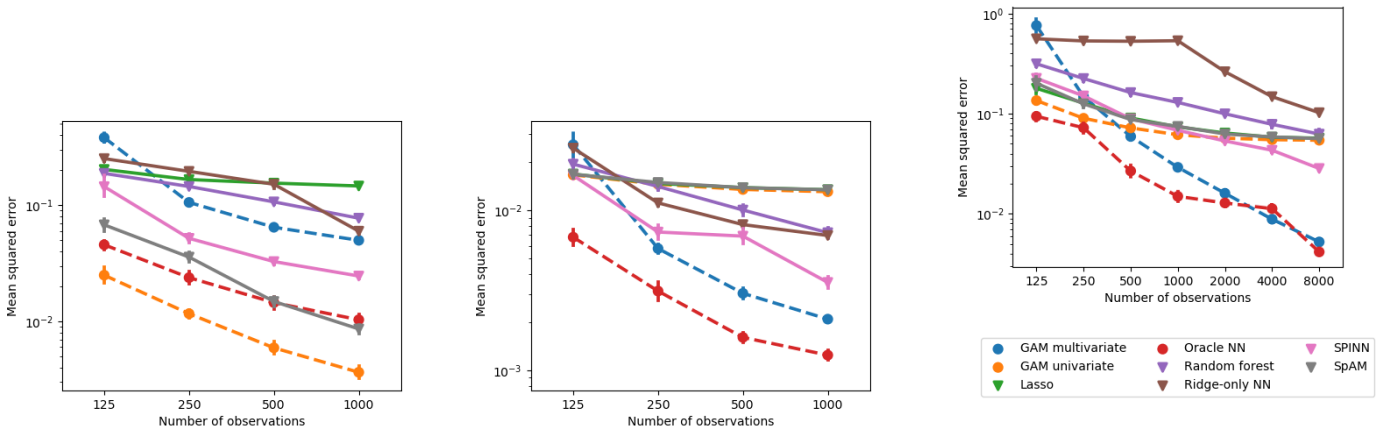
In the first scenario, we have $p = 50$ covariates and the true function is the sum of univariate functions

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

Since the true model is additive, we expect that the additive univariate oracle performs the best, followed by SpAM. As shown in Figure 3 a, 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 and random forests perform poorly in this scenario since they are unable to determine which variables are relevant.

Since the true model is an additive univariate model, a neural network that closely approximates f^* should capture this additive structure by estimating a separate network for each covariate and then output a linear combination of their outputs. As such, the nodes in the first hidden layer of this "additive" neural network should be very sparsely connected: each of these nodes should be connected to exactly a single input node. We were not able to recover this behavior in the fitted networks using SPINN due to the small sample size and because neural networks tend to prefer fitting models with interactions rather than univariate additive models. Nonetheless, the lasso did seem to encourage fitting models where the hidden nodes were only connected to a small number of input nodes. For example, in a simulation replicate with 2000 training observations where 43 features were included in the fitted model, some hidden nodes had 36 nonzero incoming edges whereas others had 43 nonzero incoming edges.



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

Figure 3: Simulation results from the three scenarios: additive univariate function with $p = 50$ (a), a complex multivariate function $p = 50$ (b), and the sum of multivariate functions with $p = 1000$ (c). The dashed lines and triangles indicate oracle models.

5.3.2 Complex multivariate model

In the second simulation, we use $p = 50$ covariates and a sparse, multivariate generative 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 3 b. Similar to results in Section 5.3.1, the performance of sparse-input neural networks more closely 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 sufficiently large training set, even the ridge-only neural network and random forests can outperform the additive univariate methods.

5.3.3 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.

The results (Figure 3 c) are a mixture of the results from the previous two simulation setups: SpAM and sparse-input neural networks perform similarly in small samples and diverge at larger sample sizes; and random forests and ridge-penalized neural networks perform poorly. These results illustrate that the utility of SPINN depends on the problem.

Since f^* is the sum of low-dimensional functions, models that ignore interactions can still perform well when the number of observations is much smaller than the number of features; SPINN is more useful in this problem as it becomes more moderate-dimensional.

6 Data analyses

We now compare SPINN to the other methods on several high- and moderate-dimensional datasets. The first task is to predict phenotypes given gene expression microarray data. The second task is to predict binding affinity between peptides and protein receptors. Employing nonparametric techniques in these problems may improve performance since the biological processes are typically believed to involve complex higher-order interactions between various components.

6.1 Predicting phenotypes from gene expression data

Here we consider two very high-dimensional problems involving gene expression data. The first dataset is a regression problem where our goal is to predict the production rate of riboflavin in *Bacillus subtilis* given gene expression profiles of $p = 4088$ genes in $n = 71$ experimental settings (Bühlmann et al. 2014). The second dataset entails predicting the molecular type of samples (NEG or not NEG) collected from $n = 128$ patients with acute lymphoblastic leukemia (Chiaretti et al. 2004, Li 2018). For each subject, $p = 12625$ gene expression levels were measured using Affymetrix human 95Av2 arrays and normalized using the robust multichip average method. Based on pre-tuning the network structure, we cross-validated over networks with one or two hidden layers where the first layer had ten nodes and the second layer had up to four nodes for the first dataset. For the second dataset, we only considered a single hidden layer where the first layer had three or ten nodes.

For both datasets, 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 methods based on the average performance across 30 random splits of the dataset (Table 1). We also evaluate how many features are included in the fitted model for the various methods. For a neural network, we consider a feature to be included in the fitted model if the norm of the weights connected that that feature is nonzero.

The top performing methods on these two datasets are sparse-input neural networks and simple linear/logistic models employing the lasso penalty. SPINN achieves the best estimated error in the first dataset and the two methods are not significantly different in the second. This is particularly impressive as these datasets are extremely high-dimensional and "common knowledge" has strongly pushed against using neural networks in such settings, as evidenced by the poor performance of the standard ridge-penalized network. These experiments suggest that SPINN is promising for analyzing gene expression datasets, particularly with the recent push to generate larger genomic datasets (Collins & Varmus 2015).

Table 1: Average performance of the different methods for predicting riboflavin production rates in *Bacillus subtilis* (top) and classification of Leukemia samples (bottom). The "# feats incl." indicates the number of features included in the fitted model. Standard error given in parentheses.

Predicting riboflavin production rates

Method	Mean squared error	# feats incl.
Sparse-input NN	0.116 (0.009)	45.1 (1.6)
Ridge-penalized NN	0.145 (0.014)	4088 (0)
SpAM	0.148 (0.015)	32.9 (2.1)
Linear model with Lasso	0.128 (0.009)	19.9 (1.26)
Random forest	0.206 (0.020)	–

Classification of Leukemia samples

Method	Classification error	# feats incl.
Sparse-input NN	0.149 (0.012)	43.7 (2.1)
Ridge-penalized NN	0.183 (0.015)	12625 (0)
SpAM	0.231 (0.015)	47.1 (4.2)
Logistic regression with Lasso	0.141 (0.011)	44.1 (2.2)
Random forest	0.179 (0.011)	–

6.2 Peptide-MHC binding affinity prediction

Here we apply our method to predict the binding affinity between peptides and class I major histocompatibility complexes (MHCs). Class I MHCs are receptors on the surface of nearly every nucleated cell in the body and bind to peptides (typically eight to eleven amino acids) derived from intracellular proteins. These peptide-MHC complexes help the immune system monitor the health of each cell and eliminate infected or tumoral cells. Peptides that cannot be bound by an individual's MHC are "invisible" to the T cell component of their immune system, thus understanding the binding affinity between the peptide and MHC is important developing vaccines and immunotherapies against cancer.

Each MHC molecule can only bind to a specific subset of peptides and the binding affinity depends on how amino acids interact. As such, the most accurate algorithms for predicting binding affinity are fully nonparametric (Andreatta & Nielsen 2016, Jurtz et al. 2017, O'Donnell et al. 2018, Boehm et al. 2018) and the start-of-the-art method employs ridge-penalized neural networks (Trolle et al. 2015, Zhao & Sher 2018). Here we apply our method to see if incorporating the sparse group lasso can further improve prediction accuracy.

Since most peptides that bind to MHC are composed of nine amino acids (AAs), we will only consider peptides of this length in this analysis. We used an "allele-specific" approach where separate predictors are trained for each MHC allele. We trained the models on quantitative binding affinity data available from `https://github.com/openvax/mhcflurry`, which was collated from the Immune Epitope database (IEDB) (Vita et al. 2015) and a published benchmark (Kim et al. 2014). For each amino acid, we generated a one-hot vector as well as the following physical features: Kidera factors (Kidera et al. 1985) and features generated in ForestMHC (Boehm et al. 2018) (hydropathy score, molar mass, and whether

Table 2: Average performance of peptide-MHC binding prediction for three HLA alleles over 30 replicates. The "# feats" indicates the number of features included in the fitted model. MSE = mean squared error. Standard error given in parentheses.

	HLA-A*01:01 $n = 3370$		HLA-B*44:02 $n = 1430$		HLA-B*08:02 $n = 487$	
Method	MSE	# feats	MSE	# feats	MSE	# feats
SPINN	**0.147** (0.002)	154 (2)	**0.193** (0.007)	116 (3)	**0.17** (0.01)	97 (3)
Ridge NN	0.249 (0.022)	297 (0)	0.268 (0.022)	297 (0)	0.25 (0.03)	297 (0)
SpAM	0.247 (0.003)	63 (1)	0.262 (0.012)	59 (2)	0.24 (0.01)	33 (2)
Lasso	0.247 (0.003)	107 (2)	0.257 (0.006)	96 (4)	0.26 (0.01)	67 (2)
Random forest	0.161 (0.003)	–	**0.191** (0.005)	–	0.21 (0.01)	–

or not the amino acid was aromatic). In total, each input amino acid was represented by a 33-dimensional vector and each peptide was represented by 297 covariates. As there are many MHC alleles, we sampled three alleles to benchmark performance: HLA-A*01:01, HLA-B*44:02, and HLA-B*08:02. These were chosen to represent datasets with different number of observations. We considered neural network architectures with one or two hidden layers, where the first layer had 45 nodes and the second layer had 5 to 20 nodes. We split the dataset into a training and test set in the same manner as before.

SPINN performed significantly better for two of the three HLA alleles and tied with random forests for allele HLA-B*44:02 (Table 2). Since the peptide-MHC binding affinity depends on interactions between amino acids, it is unsurprising that Lasso and SpAM performed poorly. Given that there was a significant improvement over the ridge-penalized neural network, we believe that incorporating a sparse group lasso penalty can significantly improve existing peptide-MHC binding affinity prediction methods.

For each HLA allele, we found that the estimated SPINN model used at least one generated feature from each position. For each position in the peptide, the nonzero features typically included a subset of the Kidera factors, some elements from the one-hot vector, and a subset of the remaining features from ForestMHC. By modeling interactions between input features, SPINN tended to employ more input features than the additive models SpAM and Lasso. Nonetheless, SPINN zeroed-out many features; even for HLA-A*01:01 which had the most observations, the fitted model only used an average of 154 of the 297 features.

7 Discussion

We have introduced the use of sparse-input neural networks as a nonparametric regression and classification 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 sparse group lasso has a prediction error that grows logarithmically with the number of features. Thus sparse-input neural networks can be effective for modeling high-dimensional data. We have also provided empirical evidence via simulation studies and data analyses to show that sparse-input neural networks can outmatch other more traditional nonparametric methods in high dimensions. Our results

show that neural networks can indeed be applied to high-dimensional datasets, as long as proper regularization is applied.

One possible direction of further research is to understand how sparsity-inducing penalties in upper-layer weights can be used to tune the size of the network and how it affects prediction accuracy. If the size of such networks can be controlled by a small number of penalty parameters, then this could potentially reduce the number of hyper-parameters that need to be tuned.

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, other methods like sparse additive models and random forests are much faster to fit. The advantage of SPINN over additive models is most apparent when modeling complex higher-order interactions drastically improves performance; the advantage of SPINN over other fully nonparametric methods is most apparent when the dataset is very high-dimensional but the number of observations is relatively small.

The code for fitting sparse-input neural networks is available at `http://github.com/jjfeng/spinn`.

Acknowledgments

We thank Frederick A. Matsen IV for helpful discussion and suggestions.

A Proofs

A.1 Proof of Theorem 1

The proof for Theorem 1 is composed of two main steps. First, we show that the excess loss of any $f_{\boldsymbol{\eta}}$ is lower bounded by a quadratic function of the distance from $\boldsymbol{\eta}$ to EQ^*. We combine this with the definition of $\hat{\boldsymbol{\eta}}$ and $\mathcal{T}_{\tilde{\lambda},T}$ to derive the result.

Using Conditions 1, 2, and 3, we lower bound the excess loss by a quadratic function. Let m^* be the number of parameters in the network excluding the ones connected to the irrelevant weights.

Lemma 1 (Quadratic lower bound)**.** *Suppose Conditions 1, 2, 3 hold. For some constant $R > 0$, suppose $\boldsymbol{\eta} \in \Theta$ satisfies*

$$\|\boldsymbol{\eta}_S - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}\|_2 \leq R \tag{25}$$

and

$$\|\boldsymbol{\theta}_{S^c}\|_1 \leq c\left(\left\|\text{other}(\boldsymbol{\eta}) - \text{other}\left(\boldsymbol{\eta}^{*(\boldsymbol{\eta})}\right)\right\|_2 + 3\sum_{j\in S}\Omega_\alpha\left(\boldsymbol{\theta}_{1,\cdot,j} - \boldsymbol{\theta}^{*(\boldsymbol{\eta})}_{1,\cdot,j}\right)\right) \tag{26}$$

for some $c > 0$. Then we have $\mathcal{E}(\boldsymbol{\eta}) \geq \|\boldsymbol{\eta}_S - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}\|_2^2 / C_0^2$ where

$$C_0^2 = \frac{1}{\epsilon_0} \vee \frac{R^2}{\chi_{\epsilon_0}}, \tag{27}$$

$$\epsilon_0 = \frac{h_{min}}{C_1 G c^3\left((1-\alpha)\sqrt{m_1|S|} + \alpha\sqrt{|S|} + \sqrt{m^*}\right)^3} \tag{28}$$

for some constant $C_1 > 0$.

Proof. For convenience, denote $d(\boldsymbol{\eta}) = \|\boldsymbol{\eta}_S - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}\|_2$. Since the gradient of the loss function is zero at all $\boldsymbol{\eta}^* \in EQ^*$, we have by Taylor expansion

$$\mathcal{E}(\boldsymbol{\eta}) = \frac{1}{2}\left(\boldsymbol{\eta} - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}\right)^\top \left[\nabla^2_\eta \mathbb{P}\ell_\eta(y, \boldsymbol{x})\right]_{\eta=\eta^{*(\eta)}} \left(\boldsymbol{\eta} - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}\right) + r_\eta \tag{29}$$

where r_η is the Lagrange remainder. By Condition 1, we have that

$$\left(\boldsymbol{\eta} - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}\right)^\top \left[\nabla^2_\eta \mathbb{P}\ell_\eta(y, \boldsymbol{x})\right]_{\eta=\eta^{*(\eta)}} \left(\boldsymbol{\eta} - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}\right) \geq h_{min} d^2(\boldsymbol{\eta}). \tag{30}$$

In addition, r_η is bounded above by

$$|r_\eta| \leq \frac{1}{6}\|\boldsymbol{\eta} - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}\|_1^3 \mathbb{P}\left[\sup_{\eta\in\Theta}\max_{j_1,j_2,j_3}\left|\frac{\delta^3\ell_\eta(y,\boldsymbol{x})}{\delta\eta_{j_1}\delta\eta_{j_2}\delta\eta_{j_3}}\right|\right] \tag{31}$$

$$\leq \frac{G}{6}\|\boldsymbol{\eta} - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}\|_1^3 \tag{32}$$

where we used Condition 3 in the last line. Moreover, if (26) holds, then

$$\|\boldsymbol{\theta}_{S^c}\|_1 \leq c\left(\left\|\text{other}(\boldsymbol{\eta}) - \text{other}\left(\boldsymbol{\eta}^{*(\eta)}\right)\right\|_2 + 3\left[(1-\alpha)\sqrt{m_1} + \alpha\right]\sqrt{|S|}\left\|\boldsymbol{\theta}_{1,\cdot,S} - \boldsymbol{\theta}^{*(\eta)}_{1,\cdot,S}\right\|_2\right) \tag{33}$$

$$\leq c\left(1 + 3(1-\alpha)\sqrt{m_1|S|} + 3\alpha\sqrt{|S|}\right) d(\boldsymbol{\eta}), \tag{34}$$

Since $\|\boldsymbol{\eta} - \boldsymbol{\eta}^{*(\eta)}\|_1 = \|\boldsymbol{\theta}_{1,\cdot,S^c}\|_1 + \|\boldsymbol{\eta}_S - \boldsymbol{\eta}^{*(\eta)}\|_1$, we can combine (32) and (34) to get

$$|r_\eta| \leq \frac{G}{6}c^3\left(1 + 3(1-\alpha)\sqrt{m_1|S|} + 3\alpha\sqrt{|S|} + \sqrt{m^*}\right)^3 d^3(\boldsymbol{\eta}) \tag{35}$$

$$\leq C_1 G c^3\left((1-\alpha)\sqrt{m_1|S|} + \alpha\sqrt{|S|} + \sqrt{m^*}\right)^3 d^3(\boldsymbol{\eta}) \tag{36}$$

for a constant C_1. Combining (30) and (36) gives us

$$\mathcal{E}(\boldsymbol{\eta}) \geq \frac{1}{2}h_{min}d^2(\boldsymbol{\eta}) - \frac{G}{6}c^3\left((1-\alpha)\sqrt{m_1|S|} + \alpha\sqrt{|S|} + \sqrt{m^*}\right)^3 d^3(\boldsymbol{\eta}).$$

Now we use Condition 2 and apply Auxiliary Lemma in Städler et al. (2010). In particular, the Auxiliary Lemma states that

> **Auxiliary Lemma in Städler et al. (2010)** Let $h : [-R, R] \mapsto [0, \infty)$ have the following properties:
>
> 1. $\forall \epsilon > 0, \exists \chi_\epsilon > 0$ such that $\inf_{\epsilon < z \leq R} h(z) \geq \chi_\epsilon$,
> 2. $\exists \Lambda > 0, C > 0$, such that $\forall z \leq R, h(z) \geq \Lambda^2 z^2 - Cz^3$.
>
> Then $\forall |z| \leq R$,
>
> $$h(z) \geq z^2/C_0^2 \tag{37}$$
>
> where
>
> $$C_0^2 = \frac{1}{\epsilon_0} \vee \frac{K_0^2}{\chi_{\epsilon_0}}, \epsilon_0 = \frac{\Lambda^2}{2C}. \tag{38}$$

To use the Auxiliary Lemma, plug in $z = d(\boldsymbol{\eta})$. □

Finally, we are ready to prove Theorem 1. In this document, we use the notation $\ell_\eta(y, \boldsymbol{x}) = \ell(y, f_\eta(\boldsymbol{x}))$.

Proof of Theorem 1. For simplicity, we'll denote $\boldsymbol{\eta}^{*(\hat{\eta})}$, the closest point in EQ^* to $\hat{\boldsymbol{\eta}}$, by $\boldsymbol{\eta}^*$. By definition of $\hat{\boldsymbol{\eta}}$,

$$\mathbb{P}_n \ell_{\hat{\eta}}(y, \boldsymbol{x}) + \lambda \sum_{j=1}^{p} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j}) \leq \mathbb{P}_n \ell_{\eta^*}(y, \boldsymbol{x}) + \lambda \sum_{j \in S} \Omega_\alpha(\boldsymbol{\theta}^*_{1,\cdot,j}).$$

Rearranging, we get

$$\mathcal{E}(\hat{\boldsymbol{\eta}}) + \lambda \sum_{j=1}^{p} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j}) \leq |(\mathbb{P}_n - \mathbb{P})(\ell_{\eta^*}(y, \boldsymbol{x}) - \ell_{\hat{\eta}}(y, \boldsymbol{x}))| + \lambda \sum_{j \in S} \Omega_\alpha(\boldsymbol{\theta}^*_{1,\cdot,j}).$$

Over the set $\mathcal{T}_{\tilde{\lambda},T}$, we have that

$$\mathcal{E}(\hat{\boldsymbol{\eta}}) + \lambda \sum_{j=1}^{p} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j}) \leq T\tilde{\lambda}\left(\tilde{\lambda} \vee \left(\|\text{other}(\hat{\boldsymbol{\eta}}) - \text{other}(\boldsymbol{\eta}^*)\|_2 + \sum_{j=1}^{p} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j})\right)\right) + \lambda \sum_{j \in S} \Omega_\alpha(\boldsymbol{\theta}^*_{1,\cdot,j}). \tag{39}$$

Now we consider three possible cases.

Case 1: $\tilde{\lambda} \geq \|\text{other}(\hat{\boldsymbol{\eta}}) - \text{other}(\boldsymbol{\eta}^*)\|_2 + \sum_{j=1}^{p} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j})$
In this case, we can rearrange (39) and apply the triangle inequality to get

$$\mathcal{E}(\hat{\boldsymbol{\eta}}) + \lambda \sum_{j \in S^c} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j}) \leq T\tilde{\lambda}^2 + \lambda \sum_{j \in S} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j}). \tag{40}$$

Since $\tilde{\lambda} \geq \sum_{j=\in S} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j})$, we have

$$\mathcal{E}(\hat{\boldsymbol{\eta}}) + \lambda \sum_{j \in S^c} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{(j)}) \leq T\tilde{\lambda}^2 + \lambda\tilde{\lambda}. \tag{41}$$

Case 2: $\tilde{\lambda} < \|\text{other}(\hat{\boldsymbol{\eta}}) - \text{other}(\boldsymbol{\eta}^*)\|_2 + \sum_{j=1}^{p} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j})$
In this case, we can rearrange (39) and apply the triangle inequality to get

$$\mathcal{E}(\hat{\boldsymbol{\eta}}) + (\lambda - T\tilde{\lambda}) \sum_{j \in S^c} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j}) \tag{42}$$

$$\leq T\tilde{\lambda}\|\text{other}(\hat{\boldsymbol{\eta}}) - \text{other}(\boldsymbol{\eta}^*)\|_2 + (\lambda + T\tilde{\lambda}) \sum_{j \in S} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j}). \tag{43}$$

We now show that (26) in Lemma 1 is satisfied. To see this, note that

$$\|\hat{\boldsymbol{\theta}}_{S^c}\|_1 \leq \frac{1}{1 - \alpha + \alpha/\sqrt{m_1}} \sum_{j \in S^c} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{(j)}). \tag{44}$$

Also, we have from (43)

$$\sum_{j \in S^c} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{(j)}) \leq \frac{T\tilde{\lambda}}{\lambda - T\tilde{\lambda}} \|\text{other}(\hat{\boldsymbol{\eta}}) - \text{other}(\boldsymbol{\eta}^*)\|_2 + \frac{\lambda + T\tilde{\lambda}}{\lambda - T\tilde{\lambda}} \sum_{j \in S} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j} - \boldsymbol{\theta}_{1,\cdot,j}) \tag{45}$$

$$\leq \|\text{other}(\hat{\boldsymbol{\eta}}) - \text{other}(\boldsymbol{\eta}^*)\|_2 + 3 \sum_{j \in S} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j} - \boldsymbol{\theta}_{1,\cdot,j}). \tag{46}$$

Using a similar argument, we can also show that

$$\|\boldsymbol{\eta}_S - \boldsymbol{\eta}^{*(\eta)}\|_2 \leq C_1(K + \max_{\eta^* \in EQ^*} \sum_{j \in S} \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j^*}))$$

for some constant $C_1 > 0$. So (25) is satisfied. Thus Case 2 satisfies all the conditions in Lemma 1 and we have

$$\mathcal{E}(\hat{\boldsymbol{\eta}}) \geq \left\|\hat{\boldsymbol{\theta}}_S - \boldsymbol{\theta}^*\right\|_2^2 / C_0^2 \tag{47}$$

where

$$C_0^2 = \frac{1}{\epsilon_0} \vee \frac{C_1^2(K + \max_{\eta^* \in EQ^*} \sum_{j \in S} \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j^*}))^2}{\chi_{\epsilon_0}}, \tag{48}$$

$$\epsilon_0 = \frac{h_{min}(1 - \alpha + \alpha/\sqrt{m_1})^3}{C_1 G\left((1-\alpha)\sqrt{m_1|S|} + \alpha\sqrt{|S|} + \sqrt{m^*}\right)^3}. \tag{49}$$

Now from (43), we have that

$$\mathcal{E}(\hat{\boldsymbol{\eta}}) + (\lambda - T\tilde{\lambda}) \sum_{j \in S^c} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j}) \leq \left(T\tilde{\lambda} + \lambda\right) \sum_{j \in S} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j}) + T\tilde{\lambda} \|\text{other}(\hat{\boldsymbol{\eta}}) - \text{other}(\boldsymbol{\eta}^*)\|_2 \tag{50}$$

$$\leq \left(T\tilde{\lambda} + \lambda\right)((1-\alpha)\sqrt{m_1} + \alpha)\sqrt{|S|}\left\|\hat{\boldsymbol{\theta}}_S - \boldsymbol{\theta}^*\right\|_2 \tag{51}$$

$$\leq \frac{1}{2}\left[\left(T\tilde{\lambda} + \lambda\right)^2((1-\alpha)\sqrt{m_1} + \alpha)^2 |S| C_0^2 + \frac{1}{C_0^2}\left\|\hat{\boldsymbol{\theta}}_S - \boldsymbol{\theta}^*\right\|_2^2\right] \tag{52}$$

Finally, plugging in (47), we have that

$$\frac{1}{2}\mathcal{E}(\hat{\boldsymbol{\eta}}) + (\lambda - T\tilde{\lambda}) \sum_{j \in S^c} \Omega_\alpha(\hat{\boldsymbol{\theta}}_{1,\cdot,j}) \leq \frac{1}{2}\left(T\tilde{\lambda} + \lambda\right)^2((1-\alpha)\sqrt{m_1} + \alpha)^2 |S| C_0^2. \tag{53}$$

□

A.2 Proof of Theorem 2

Next we use empirical process techniques to prove Theorem 2 where we will measure complexity of function classes using metric entropy (van der Vaart & Wellner 1996). Let the u-entropy of a function class $\mathcal{G}$ with respect to the norm $\|\cdot\|$ be denoted $H(u, \mathcal{G}, \|\cdot\|)$, which is equal to the log of its u-covering number $N(u, \mathcal{G}, \|\cdot\|)$.

Define the empirical process term

$$v_n(\boldsymbol{\eta}) = (\mathbb{P}_n - \mathbb{P})\ell_\eta(y, \boldsymbol{x}). \tag{54}$$

The basic idea of the proof is to split $v_n(\boldsymbol{\eta})$ into a truncated component $v_n^{\text{trunc}}(\boldsymbol{\eta})$ and a remainder term $v_n - v_n^{\text{trunc}}(\boldsymbol{\eta})$, where

$$v_n^{\text{trunc}}(\boldsymbol{\eta}) = (\mathbb{P}_n - \mathbb{P})\left[\ell_\eta(y, \boldsymbol{x}) 1\left\{G(y) \leq M_n\right\}\right] \tag{55}$$

for some truncation function $G(\cdot)$ and constant $M_n > 0$ that can grow with n. We will later show that choosing $M_n = O_p(\sqrt{\log n})$ is appropriate for our convergence rate bounds. We then control the truncated empirical process and the remainder separately.

The function $G(\cdot)$ is used to truncate the gradient of the loss function with respect to the value of the nodes in the first hidden layer as well as the parameters for the upper layers, i.e. those above the first hidden layer. Let $\boldsymbol{\eta}_{\text{upper}}$ denote the parameters for the upper neural network. Then $f_{\boldsymbol{\eta}_{\text{upper}}}$ will denote the output of this separate neural network. Then define

$$\tilde{\ell}(y, z, \boldsymbol{\eta}_{\text{upper}}) = \ell(y, f_{\boldsymbol{\eta}_{\text{upper}}}(z)). \tag{56}$$

Then define $G(\cdot)$ as the function below.

Condition 5. *The gradient of the loss function is bounded above by*

$$\sup_{\eta \in \Theta, x \in \mathcal{X}} \left\| \nabla_{\boldsymbol{z}, \boldsymbol{\eta}_{\text{upper}}} \tilde{\ell}(y, \boldsymbol{z}, \boldsymbol{\eta}_{\text{upper}}) \Big|_{z = \psi(\boldsymbol{\theta}_1^\top \boldsymbol{x} + t)} \right\|_\infty \leq G(y) \leq C_0(|y| + C_1). \tag{57}$$

Here G and the constants $C_0, C_1 > 0$ only depend on the number of layers and nodes in the upper layers. It does not depend on p.

It is easy to show that the condition holds when ℓ is the mean squared error loss or the logistic loss. Throughout the proofs, we will work with a general $G(y)$ until the end where we must specialize to the particular $G(y)$ in the classification/regression settings.

For the rest of this document, c_i will denote some positive constant that can depend on K and X_{max}.

To control the truncated empirical process, we begin with bounding the entropy of the function class

$$\mathcal{G}_r = \left\{ \left\{ g_\eta(y, \boldsymbol{x}) \equiv \ell_\eta(y, \boldsymbol{x}) - \ell_{\eta^{*(\eta)}}(y, \boldsymbol{x}) \right\} 1\left\{G(y) \leq M_n\right\} : \boldsymbol{\eta} \in \Theta_r \right\}$$

where $r > 0$ and

$$\Theta_r = \left\{ \eta \in \Theta : \sum_{j=1}^{p} \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j} - \boldsymbol{\theta}_{1,\cdot,j}^{*(\boldsymbol{\eta})}) + \| \text{other}(\boldsymbol{\eta}) - \text{other}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})}) \|_2 \leq r \right\}. \tag{58}$$

For a given set of n observations $\{\boldsymbol{x}_i, y_i\}$ for $i = 1, ..., n$, define the empirical norm

$$\|g\|_{\mathbb{P}_n} = \left(\frac{1}{n} \sum_{i=1}^{n} g^2(y_i, \boldsymbol{x}_i) \right)^{1/2}$$

The following lemma bounds the entropy of $\mathcal{G}_r$ with respect to $\|\cdot\|_{\mathbb{P}_n}$.

Lemma 2. *Suppose Condition 5 holds and EQ^* contains Q equivalence classes. For any $r, M_n > 0$, the following holds for all $u > 0$:*

$$H\left(u, \mathcal{G}_r, \|\cdot\|_{\mathbb{P}_n}\right) \leq c_0 \left[\log Q + m_{\text{other}} \log\left(\frac{r M_n \sqrt{m_{\text{other}}} + u}{u}\right)\right. \tag{59}$$

$$\left. + m_1 \left(\frac{r X_{max} M_n \sqrt{m_1}}{c_{\alpha,m_1} u}\right)^2 \log\left(1 + p\left(\frac{c_{\alpha,m_1} u}{r X_{max} M_n \sqrt{m_1}}\right)^2\right)\right] \tag{60}$$

where $c_{\alpha,m_1} = 1 - \alpha + \alpha/\sqrt{m_1}$.

Proof. To bound the entropy of $\mathcal{G}_r$, we partition $\mathcal{G}_r$ and bound the entropy of each partition. In particular, let Ξ be the subset of EQ^* such that no two $\boldsymbol{\eta}^*, \boldsymbol{\eta}^{*'} \in \Xi$ parameterize equivalent neural networks, e.g. Ξ is composed of a representative member from each of the Q equivalence classes. Then we partition $\mathcal{G}_r$ into

$$\mathcal{G}_r \subseteq \cup_{\boldsymbol{\eta}^* \in \Xi} \mathcal{G}_{r,\boldsymbol{\eta}^*} \tag{61}$$

where $\mathcal{G}_{r,\boldsymbol{\eta}^*} = \left\{g_{\boldsymbol{\eta}} : \sum_{j=1}^p \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j}) + \|\text{other}(\boldsymbol{\eta}) - \text{other}(\boldsymbol{\eta}^*)\|_2 \leq r\right\}$.

We can partition $\mathcal{G}_r$ according to (61) due to the following claim:

Claim For any $\boldsymbol{\eta}^* \in EQ^*$, let

$$\Theta_{r,EQ(\boldsymbol{\eta}^*)} = \left\{\boldsymbol{\eta} \in \Theta_r : \boldsymbol{\eta}^{*(\boldsymbol{\eta})} \in EQ(\boldsymbol{\eta}^*)\right\} \tag{62}$$

$$\Theta_{r,\boldsymbol{\eta}^*} = \left\{\boldsymbol{\eta} \in \Theta_r : \sum_{j=1}^p \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j}) + \|\text{other}(\boldsymbol{\eta}) - \text{other}(\boldsymbol{\eta}^*)\|_2 \leq r\right\}, \tag{63}$$

we have

$$\left\{g_\eta(y, \boldsymbol{x}) : \boldsymbol{\eta} \in \Theta_{r,EQ(\boldsymbol{\eta}^*)}\right\} \subseteq \left\{g_\eta(y, \boldsymbol{x}) : \boldsymbol{\eta} \in \Theta_{r,\boldsymbol{\eta}^*}\right\}.$$

Proof. Consider any $\boldsymbol{\eta} \in \Theta_{r,EQ(\boldsymbol{\eta}^*)}$. Define π as a function mapping $\boldsymbol{\eta}^{*(\boldsymbol{\eta})}$ to $\boldsymbol{\eta}^*$ via permutation/sign-flip, i.e. $\pi(\boldsymbol{\eta}^{*(\eta)}) = \boldsymbol{\eta}^*$ Letting $\tilde{\boldsymbol{\eta}} = \pi(\boldsymbol{\eta})$, we have $\pi(\boldsymbol{\eta}) - \boldsymbol{\eta}^* = \boldsymbol{\eta} - \boldsymbol{\eta}^{*(\boldsymbol{\eta})}$ and so

$$\sum_{j=1}^p \Omega_\alpha(\tilde{\boldsymbol{\theta}}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j}) + \|\text{other}(\tilde{\boldsymbol{\eta}}) - \text{other}(\boldsymbol{\eta}^*)\|_2 \leq r.$$

Thus every element in $\Theta_{r,EQ(\boldsymbol{\eta}^*)}$ can be mapped to an element in $\Theta_{r,\boldsymbol{\eta}^*}$ that is the same function g_η. □

Therefore let us bound $N\left(u, \mathcal{G}_{r,\boldsymbol{\eta}^*}, \|\cdot\|_{\mathbb{P}_n}\right)$ for some $\boldsymbol{\eta}^* \in \Xi$. Consider any $\boldsymbol{\eta}, \boldsymbol{\eta}' \in \Theta_{r,\boldsymbol{\eta}^*}$. By the Mean Value Theorem and Condition 5, we have

$$|g_\eta(y, \boldsymbol{x}) - g_{\eta'}(y, \boldsymbol{x})| = |\ell_\eta(y, \boldsymbol{x}) - \ell_{\eta'}(y, \boldsymbol{x})| \, 1\{G(y) \leq M_n\}$$

$$\leq M_n \left(\left\|(\boldsymbol{\theta} - \boldsymbol{\theta}')^\top \boldsymbol{x}\right\|_1 + \left\|\text{other}(\boldsymbol{\eta}) - \text{other}\left(\boldsymbol{\eta}'\right)\right\|_1\right)$$

Squaring both sides and applying Cauchy Schwarz, we get

$$|g_\eta(y, \boldsymbol{x}) - g_{\eta'}(y, \boldsymbol{x})|^2 \leq 4 M_n^2 \left(m_1 \left\|(\boldsymbol{\theta} - \boldsymbol{\theta}')^\top \boldsymbol{x}\right\|_2^2 + m_{\text{other}} \left\|\text{other}(\boldsymbol{\eta}) - \text{other}\left(\boldsymbol{\eta}'\right)\right\|_2^2\right) \tag{64}$$

Therefore to bound the entropy of $\mathcal{G}_{r,\boldsymbol{\eta}^*}$, it suffices to bound the entropy of

$$\mathcal{J}_{r1} \coloneqq \{\boldsymbol{\beta} \in \mathbb{R}^{m_{\text{other}}} : \|\boldsymbol{\beta} - \text{other}(\boldsymbol{\eta}^*)\|_2 \leq r\}$$

and

$$\mathcal{J}_{r2} \coloneqq \left\{ \boldsymbol{\theta}^\top \boldsymbol{x} : \sum_{j=1}^{p} \|\boldsymbol{\theta}_{1,\cdot,j} - \boldsymbol{\theta}^*_{1,\cdot,j}\|_1 \leq \frac{r}{1-\alpha+\alpha/\sqrt{m_1}} \right\}.$$

The entropy of a ball with radius r in $\mathbb{R}^{\text{other}}$ is for all $u \geq 0$,

$$H\left(u, \mathcal{J}_{r1}, \|\cdot\|_2\right) \leq m_{\text{other}} \log\left(\frac{4r+u}{u}\right).$$

Applying a slight modification of the bound in Lemma 2.6.11 in van der Vaart & Wellner (1996), we get the following entropy bound (for all $u \geq 0$)

$$H\left(u, \mathcal{J}_{r2}, \|\cdot\|_{P_n}\right) \leq c_0 m_1 \left(\frac{r X_{max}}{(1-\alpha+\alpha/\sqrt{m_1})u}\right)^2 \log\left(1 + p\left(\frac{(1-\alpha+\alpha/\sqrt{m_1})u}{r X_{max}}\right)^2\right).$$

Putting these bounds together, the entropy of $\mathcal{G}_{r,\boldsymbol{\eta}^*}$ is bounded above for all $u \geq 0$ by

$$\begin{aligned} H\left(u, \mathcal{G}_{r,\eta_0}, \|\cdot\|_{\mathbb{P}_n}\right) &\leq H\left(\frac{u}{2M_n\sqrt{m_{\text{other}}}}, \mathcal{J}_{r1}, \|\cdot\|_2\right) + H\left(\frac{u}{2M_n\sqrt{m_1}}, \mathcal{J}_{r2}, \|\cdot\|_{P_n}\right) \\ &\leq m_{\text{other}} \log\left(\frac{8rM_n\sqrt{m_{\text{other}}}+u}{u}\right) \\ &\quad + c_0 m_1 \left(\frac{2rX_{max}M_n\sqrt{m_1}}{(1-\alpha+\alpha/\sqrt{m_1})u}\right)^2 \log\left(1 + p\left(\frac{(1-\alpha+\alpha/\sqrt{m_1})u}{2rX_{max}M_n\sqrt{m_1}}\right)^2\right). \end{aligned}$$

Because we have Q equivalence classes in EQ^*, we must add a $\log Q$ term, to attain our final bound. □

Next we need to show that the symmetrized truncated empirical process term is small with high probability. We use the Rademacher random variables W, which are defined to have distribution $\Pr(W=1) = \Pr(W=-1) = 0.5$.

Lemma 3. *Assume the same conditions as Lemma 2. Let $W_1, ..., W_n$ be n independent Rademacher random variables. Let*

$$\delta = c_3 M_n c_{\alpha,m,2} \left(\sqrt{\log Q} + \frac{X_{max}}{c_{\alpha,m_1}} \sqrt{\log\left(\frac{p c^2_{\alpha,m,2}}{m_1 X^2_{max}} + 1\right)} \log\left(n M_n c_{\alpha,m,2}\right)\right) \tag{65}$$

where $c_{\alpha,m,2} = m_1 + \sqrt{m_{\text{other}} K} + \frac{X_{max}}{c_{\alpha,m_1}}$. Then for fixed $(\boldsymbol{x}_i, y_i)$ for $i = 1, ..., n$, we have for all $r > 0$ and $T \geq 1$

$$\begin{aligned} &\Pr\left(\sup_{\eta \in \Theta_r} \left| \frac{1}{n} \sum_{i=1}^{n} W_i \left[\ell_\eta\left(y_i, \boldsymbol{x}_i\right) - \ell_{\eta^*(\eta)}\left(y_i, \boldsymbol{x}_i\right)\right] \right| \geq Tr\delta/\sqrt{n}\right) \\ &\leq c_2 \exp\left(-c_3 T^2 \left(\sqrt{\log Q} + \frac{X_{max}}{c_{\alpha,m_1}} \sqrt{\log\left(\frac{p c^2_{\alpha,m,2}}{m_1 X^2_{max}} + 1\right)} \log\left(n M_n c_{\alpha,m,2}\right)\right)^2 (r^2 \vee 1)\right). \end{aligned}$$

Taylor - expansion of composite - function.

Proof. We apply Lemma 3.2 in van de Geer (2000). First we check all the conditions are satisfied.

Using Condition 5 and Taylor expansion, we can show that for any $\boldsymbol{\eta}, \boldsymbol{\eta}' \in \Theta_r$

$$|\ell_\eta(y, \boldsymbol{x}) - \ell_{\eta'}(y, \boldsymbol{x})| \tag{66}$$

$$\leq c_0 G(y) \left[\left(m_1 + \sqrt{m_{\text{other}} K} \right) \wedge \left(\frac{X_{max}}{c_{\alpha,m_1}} \sum_{j=1}^{p} \Omega_\alpha(\theta_{1,\cdot,j} - \theta'_{1,\cdot,j}) + \| \text{other}(\boldsymbol{\eta}) - \text{other}(\boldsymbol{\eta}') \|_1 \right) \right] \tag{67}$$

for the same c_{α,m_1} defined in Lemma 2. Thus we have

$$\sup_{\eta \in \Theta_r} \frac{1}{n} \sum_{i=1}^{n} |\ell_\eta(y_i, \boldsymbol{x}_i) - \ell_{\eta'}(y_i, \boldsymbol{x}_i)|^2 \, 1\{G(y_i) \leq M_n\} \tag{68}$$

$$\leq c_1 M_n^2 \left(m_1 + \sqrt{m_{\text{other}} K} + \frac{X_{max}}{c_{\alpha,m_1}} \right)^2 (r^2 \wedge 1) \tag{69}$$

Set R_n^2 equal to the right hand side of (69) and let $\tilde{R}_n^2 = c_1 M_n^2 \left(m_1 + \sqrt{m_{\text{other}} K} + \frac{X_{max}}{c_{\alpha,m_1}} \right)^2 r^2$. Then we can bound Dudley's integral in Lemma 3.2 as follows

$$\begin{aligned}
&\int_{r/n}^{R_n} H^{1/2}(u, \mathcal{G}_r, \|\cdot\|_n) \, du \\
&\leq \int_{r/n}^{\tilde{R}_n} H^{1/2}(u, \mathcal{G}_r, \|\cdot\|_n) \, du \\
&\leq c_2 r M_n c_{\alpha,m,2} \left(\sqrt{\log Q} + \frac{X_{max}}{c_{\alpha,m_1}} \sqrt{\log\left(\frac{p c_{\alpha,m,2}^2}{m_1 X_{\max}^2} + 1 \right)} \log(n M_n c_{\alpha,m,2}) \right),
\end{aligned}$$

where $c_{\alpha,m,2} = m_1 + \sqrt{m_{\text{other}} K} + \frac{X_{max}}{c_{\alpha,m_1}}$.

Let $\delta = c_3 M_n c_{\alpha,m,2} \left(\sqrt{\log Q} + \frac{X_{max}}{c_{\alpha,m_1}} \sqrt{\log\left(\frac{p c_{\alpha,m,2}^2}{m_1 X_{\max}^2} + 1 \right)} \log(n M_n c_{\alpha,m,2}) \right)$. Then for any $T \geq 1$, we have

$$\begin{aligned}
&\Pr\left(\sup_{\eta \in \Theta_r} \left| \frac{1}{n} \sum_{i=1}^{n} W_i \left[\ell_\eta(y_i, \boldsymbol{x}_i) - \ell_{\eta^*(\eta)}(y_i, \boldsymbol{x}_i) \right] \right| \geq T r \delta / \sqrt{n} \right) \\
&\leq c_4 \exp\left(-c_4 \frac{T^2 r^2 \delta^2}{R_n^2} \right) \\
&= c_4 \exp\left(-c_4 T^2 \left(\sqrt{\log Q} + \frac{X_{max}}{c_{\alpha,m_1}} \sqrt{\log\left(\frac{p c_{\alpha,m,2}^2}{m_1 X_{\max}^2} + 1 \right)} \log(n M_n c_{\alpha,m,2}) \right)^2 (r^2 \vee 1) \right).
\end{aligned}$$

□

Using Lemma 3 above, combined with a slight modification to symmetrization Corollary 3.4 in van de Geer (2000), one can bound the difference of the truncated empirical processes over random observations $(\boldsymbol{x}_i, y_i)$.

Corollary 1. *Suppose the same assumptions hold as in Lemma 2. Let δ be defined as in (65). Then for all $r > 0$ and $T \geq 1$, the probability of that*

$$\sup_{\eta \in \Theta_r} \left| v_n^{trunc}(\boldsymbol{\eta}) - v_n^{trunc}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})}) \right| \geq c_3 T r \delta / \sqrt{n}$$

holds for random $(\boldsymbol{x}_i, y_i)$ for $i = 1, ..., n$ holds with probability no greater than

$$c_4 \exp\left(-c_5 T^2 \left(\sqrt{\log Q} + \frac{X_{max}}{c_{\alpha,m_1}} \sqrt{\log\left(\frac{p c_{\alpha,m,2}^2}{m_1 X_{max}^2} + 1 \right) \log\left(n M_n c_{\alpha,m,2} \right)} \right)^2 (r^2 \vee 1) \right).$$

Now we are ready to bound the scaled truncated empirical process over the entire parameter space Θ.

Lemma 4. *Suppose the same assumptions as Lemma 2. Let $\tilde{\lambda} = c_9 \delta / \sqrt{n}$ for δ defined in (65). Then for any $T \geq 1$, we have*

$$\Pr\left(\sup_{\eta \in \Theta} \frac{\left| v_n^{trunc}(\boldsymbol{\eta}) - v_n^{trunc}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})}) \right|}{\left(\sum_{j=1}^p \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j} - \boldsymbol{\theta}_{1,\cdot,j}^{*(\boldsymbol{\eta})}) + \| \text{other}(\boldsymbol{\eta}) - \text{other}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})}) \|_2 \right) \vee \tilde{\lambda}} \geq T c_6 \tilde{\lambda} \right)$$
$$\leq c_7 \log n \exp\left(-c_8 T^2 \left(\sqrt{\log Q} + \frac{X_{max}}{c_{\alpha,m_1}} \sqrt{\log\left(\frac{p c_{\alpha,m,2}^2}{m_1 X_{max}^2} + 1 \right) \log\left(n M_n c_{\alpha,m,2} \right)} \right)^2 \right).$$

Proof. We use a peeling argument by partitioning Θ into

$$\Theta = \left[\cup_{j=J}^{\infty} \Theta_j \right] \cup \Theta_{J-1}$$

where $J = \max\left\{ j : 2^j \geq \tilde{\lambda} \right\} = -c_6 \log(n)$ and

$$\Theta_j = \left\{ \boldsymbol{\eta} \in \Theta : 2^{j-1} < \sum_{j=1}^p \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j} - \boldsymbol{\theta}_{1,\cdot,j}^{*(\boldsymbol{\eta})}) + \| \text{other}(\boldsymbol{\eta}) - \text{other}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})}) \|_2 \leq 2^j \right\} \quad \forall j = J, ..., \infty$$

and

$$\Theta_{J-1} = \left\{ \boldsymbol{\eta} \in \Theta : \sum_{j=1}^p \Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j} - \boldsymbol{\theta}_{1,\cdot,j}^{*(\boldsymbol{\eta})}) + \| \text{other}(\boldsymbol{\eta}) - \text{other}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})}) \|_2 \leq 2^{J-1} \right\}.$$

Then using a peeling argument, we have

$$
\begin{aligned}
&\Pr\left(\sup_{\eta\in\Theta}\frac{|v_n^{\text{trunc}}(\boldsymbol{\eta})-v_n^{\text{trunc}}(\boldsymbol{\eta}_0)|}{\left(\sum_{j=1}^p\Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j}-\boldsymbol{\theta}_{1,\cdot,j}^{*(\boldsymbol{\eta})})+\|\operatorname{other}(\boldsymbol{\eta})-\operatorname{other}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})})\|_2\right)\vee\tilde{\lambda}}\geq T\tilde{\lambda}\right)\\
&\leq\sum_{j=J-1}^{\infty}\Pr\left(\sup_{\eta\in\Theta_j}\frac{|v_n^{\text{trunc}}(\boldsymbol{\eta})-v_n^{\text{trunc}}(\boldsymbol{\eta}_0)|}{\left(\sum_{j=1}^p\Omega_\alpha(\boldsymbol{\theta}_{1,\cdot,j}-\boldsymbol{\theta}_{1,\cdot,j}^{*(\boldsymbol{\eta})})+\|\operatorname{other}(\boldsymbol{\eta})-\operatorname{other}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})})\|_2\right)\vee\tilde{\lambda}}\geq T\tilde{\lambda}\right)\\
&\leq c_7J\exp\left(-c_8T^2\left(\sqrt{\log Q}+\frac{X_{max}}{c_{\alpha,m_1}}\sqrt{\log\left(\frac{pc_{\alpha,m,2}^2}{m_1X_{\max}^2}+1\right)\log\left(nM_nc_{\alpha,m,2}\right)}\right)^2\right)\\
&+\sum_{j=1}^{\infty}c_9\exp\left(-c_82^{2j}T^2\left(\sqrt{\log Q}+\frac{X_{max}}{c_{\alpha,m_1}}\sqrt{\log\left(\frac{pc_{\alpha,m,2}^2}{m_1X_{\max}^2}+1\right)\log\left(nM_nc_{\alpha,m,2}\right)}\right)^2\right)\\
&\leq c_{10}\log n\exp\left(-c_{11}T^2\left(\sqrt{\log Q}+\frac{X_{max}}{c_{\alpha,m_1}}\sqrt{\log\left(\frac{pc_{\alpha,m,2}^2}{m_1X_{\max}^2}+1\right)\log\left(nM_nc_{\alpha,m,2}\right)}\right)^2\right).
\end{aligned}
$$

□

The last step to proving Theorem 2 is to control the remainder term

$$v_n^{\text{rem}}(\boldsymbol{\eta})=v_n(\boldsymbol{\eta})-v_n^{\text{trunc}}(\boldsymbol{\eta}). \tag{70}$$

To bound the remainder term, we incorporate the fact that $y-f^*$ is a sub-gaussian random variable.

Lemma 5. *Suppose ϵ are independent sub-gaussian random variables. Suppose Condition 5 holds. For any $\kappa\geq 1$, we can choose some $c>0$ such that for $M_n>0$, we have*

$$\Pr\left(\frac{\left|v_n^{\text{rem}}(\boldsymbol{\eta})-v_n^{\text{rem}}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})})\right|}{\sum_{j=1}^p\Omega_\alpha(\boldsymbol{\theta}_{(j)}-\boldsymbol{\theta}_{0,(j)}^{(\boldsymbol{\eta})})+\|\operatorname{other}(\boldsymbol{\eta})-\operatorname{other}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})})\|_2}\geq\frac{c_{13}\delta}{\sqrt{n}}\right)\leq O_p\left(\frac{c_{\alpha,m,2}\exp\left(-M_n^2\right)}{\delta}\right) \tag{71}$$

where δ was defined in (65).

Proof. To bound this probability, first we find an upper bound that has a tail behavior that is easier to bound. By Taylor expansion, we have that

$$
\begin{aligned}
&\frac{\left|\ell_\eta(y,\boldsymbol{x})-\ell_{\eta^{*(\boldsymbol{\eta})}}(y,\boldsymbol{x})\right|1\left\{G(y)>M_n\right\}}{\sum_{j=1}^p\Omega_\alpha(\boldsymbol{\theta}_{(j)}-\boldsymbol{\theta}_{0,(j)}^{(\boldsymbol{\eta})})+\|\operatorname{other}(\boldsymbol{\eta})-\operatorname{other}(\boldsymbol{\eta}^{*(\boldsymbol{\eta})})\|_2}\\
&\leq C_0\left(|y|+C_1\right)1\left\{C_0\left(|y|+C_1\right)\right\}\left(\frac{X_{\max}}{c_{\alpha,m_1}}+\sqrt{m_{\text{other}}}\right).
\end{aligned}
\tag{72}
$$

Then we can upper bound (71) by

$$\Pr\left(C_0\left(\frac{X_{\max}}{c_{\alpha,m_1}}+\sqrt{m_{\text{other}}}\right)(\mathbb{P}_n-\mathbb{P})\left[(|y_i|+C_1)\,1\left\{C_0\left(|y_i|+C_1\right)\geq M_n\right\}\right]\geq\delta\right) \tag{73}$$

The probability in (73) can be bounded using Markov's inequality and the fact that sub-gaussian random variables Z satisfy

$$E\left[|Z|\,1\left\{|Z|\geq M\right\}\right]\leq C'\exp\left(-c'M^2\right) \tag{74}$$

for constants $C', c' > 0$ that only depend on the sub-gaussian parameters. □

Finally we prove Theorem 2 by setting $M_n = O_p(\sqrt{\log n})$ and combining the results in Lemmas 4 and 5.

References

Alvarez, J. M. & Salzmann, M. (2016), Learning the number of neurons in deep networks, *in* 'Advances in Neural Information Processing Systems', pp. 2270–2278.

Andreatta, M. & Nielsen, M. (2016), 'Gapped sequence alignment using artificial neural networks: application to the MHC class I system', *Bioinformatics* **32**(4), 511–517.

Anthony, M. & Bartlett, P. L. (2009), *Neural Network Learning: Theoretical Foundations*, Cambridge University Press.

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

Barron, A. R. (1993), 'Universal approximation bounds for superpositions of a sigmoidal function', *IEEE Transactions on Information theory* **39**(3), 930–945.

Bartlett, P. L. (1998), '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.

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

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

Boehm, K. M., Bhinder, B., Raja, V. J., Dephoure, N. & Elemento, O. (2018), Predicting peptide presentation by major histocompatibility complex class I using one million peptides.

Breiman, L. (2001), 'Random forests', *Mach. Learn.* **45**(1), 5–32.

Breiman, L., Friedman, J., Stone, C. J. & Olshen, R. A. (1984), *Classification and regression trees*, CRC press.

Bühlmann, P., Kalisch, M. & Meier, L. (2014), 'High-dimensional statistics with a view toward applications in biology', *Annual Review of Statistics and Its Application* **1**, 255–278.

Bühlmann, P. & Van De Geer, S. (2011), *Statistics for high-dimensional data: methods, theory and applications*, Springer Science & Business Media.

Chiaretti, S., Li, X., Gentleman, R., Vitale, A., Vignetti, M., Mandelli, F., Ritz, J. & Foa, R. (2004), 'Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival', *Blood* **103**(7), 2771–2778.

Collins, F. S. & Varmus, H. (2015), 'A new initiative on precision medicine', *N. Engl. J. Med.* **372**(9), 793–795.

Cristianini, N. & Shawe-Taylor, J. (2000), *An introduction to support vector machines and other kernel-based learning methods*, Cambridge university press.

Daubechies, I., Defrise, M. & De Mol, C. (2004), 'An iterative thresholding algorithm for linear inverse problems with a sparsity constraint', *Communications on pure and applied mathematics* **57**(11), 1413–1457.

Fefferman, C. (1994), 'Reconstructing a neural net from its output', *Revista Matemática Iberoamericana* **10**(3), 507–555.

Ghadimi, S. & Lan, G. (2016), 'Accelerated gradient methods for nonconvex nonlinear and stochastic programming', *Mathematical Programming* **156**(1-2), 59–99.

Gong, P., Zhang, C., Lu, Z., Huang, J. & Ye, J. (2013), A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems., *in* 'ICML (2)', pp. 37–45.

Graves, A., Mohamed, A.-r. & Hinton, G. (2013), Speech recognition with deep recurrent neural networks, *in* 'Acoustics, speech and signal processing (icassp), 2013 ieee international conference on', IEEE, pp. 6645–6649.

Hofmann, T., Schölkopf, B. & Smola, A. J. (2008), 'Kernel methods in machine learning', *The Annals of Statistics* **36**(3), 1171–1220.
URL: *http://www.jstor.org/stable/25464664*

Jurtz, V., Paul, S., Andreatta, M., Marcatili, P., Peters, B. & Nielsen, M. (2017), 'NetMHCpan-4.0: Improved Peptide-MHC class I interaction predictions integrating eluted ligand and peptide binding affinity data', *J. Immunol.* **199**(9), 3360–3368.

Kidera, A., Konishi, Y., Oka, M., Ooi, T. & Scheraga, H. A. (1985), 'Statistical analysis of the physical properties of the 20 naturally occurring amino acids', *J. Protein Chem.* **4**(1), 23–55.

Kim, Y., Sidney, J., Buus, S., Sette, A., Nielsen, M. & Peters, B. (2014), 'Dataset size and composition impact the reliability of performance benchmarks for peptide-MHC binding predictions', *BMC Bioinformatics* **15**, 241.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

Leshno, M., Lin, V. Y., Pinkus, A. & Schocken, S. (1993), 'Multilayer feedforward networks with a nonpolynomial activation function can approximate any function', *Neural Netw.* **6**(6), 861–867.

Li, X. (2018), 'All: A data package', *R package version 1.24.0* .

Meier, L., Van de Geer, S., Bühlmann, P. et al. (2009), 'High-dimensional additive modeling', *The Annals of Statistics* **37**(6B), 3779–3821.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013), Distributed representations of words and phrases and their compositionality, *in* 'Advances in neural information processing systems', pp. 3111–3119.

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

Nelder, J. A. & Mead, R. (1965), 'A simplex method for function minimization', *The computer journal* **7**(4), 308–313.

Nesterov, Y. (2004), *Introductory lectures on convex optimization: A basic course*, Vol. 87, Applied Optimization.

O'Donnell, T. J., Rubinsteyn, A., Bonsack, M., Riemer, A. B., Laserson, U. & Hammerbacher, J. (2018), 'MHCflurry: Open-Source class I MHC binding affinity prediction', *Cell Syst* **7**(1), 129–132.e4.

Ravikumar, P., Liu, H., Lafferty, J. & Wasserman, L. (2007), Spam: Sparse additive models, *in* 'Proceedings of the 20th International Conference on Neural Information Processing Systems', Curran Associates Inc., pp. 1201–1208.

Scardapane, S., Comminiello, D., Hussain, A. & Uncini, A. (2016), 'Group sparse regularization for deep neural networks', *arXiv preprint arXiv:1607.00485* .

Simon, N., Friedman, J., Hastie, T. & Tibshirani, R. (2013), 'A sparse-group lasso', *Journal of Computational and Graphical Statistics* **22**(2), 231–245.

Snoek, J., Larochelle, H. & Adams, R. P. (2012), Practical bayesian optimization of machine learning algorithms, *in* 'Advances in neural information processing systems', pp. 2951–2959.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., Potts, C. et al. (2013), Recursive deep models for semantic compositionality over a sentiment treebank, *in* 'Proceedings of the conference on empirical methods in natural language processing (EMNLP)', Vol. 1631, p. 1642.

Städler, N., Bühlmann, P. & Van De Geer, S. (2010), 'L1-penalization for mixture regression models', *Test* **19**(2), 209–256.

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

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 1–9.

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

Trolle, T., Metushi, I. G., Greenbaum, J. A., Kim, Y., Sidney, J., Lund, O., Sette, A., Peters, B. & Nielsen, M. (2015), 'Automated benchmarking of peptide-MHC class I binding predictions', *Bioinformatics* **31**(13), 2174–2181.

van de Geer, S. A. (2000), *Empirical Processes in M-estimation*, Vol. 6, Cambridge university press.

van der Vaart, A. W. & Wellner, J. A. (1996), *Weak convergence and empirical processes: with applications to statistics*, Springer Science & Business Media.

Vita, R., Overton, J. A., Greenbaum, J. A., Ponomarenko, J., Clark, J. D., Cantrell, J. R., Wheeler, D. K., Gabbard, J. L., Hix, D., Sette, A. & Peters, B. (2015), 'The immune epitope database (IEDB) 3.0', *Nucleic Acids Res.* **43**(Database issue), D405–12.

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

Yuan, M. & Lin, Y. (2006), 'Model selection and estimation in regression with grouped variables', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **68**(1), 49–67.

Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. (2016), 'Understanding deep learning requires rethinking generalization', *arXiv preprint arXiv:1611.03530* .

Zhao, W. & Sher, X. (2018), 'Systematically benchmarking peptide-MHC binding predictors: From synthetic to naturally processed epitopes', *PLoS Comput. Biol.* **14**(11), e1006457.