Richness of Training Data Does Not Suffice: Robustness of Neural Networks Requires Richness of Hidden-Layer Activations

Kamil Nar [1] **S. Shankar Sastry** [1]

Abstract

To learn the parameters of a model with an online learning algorithm accurately, the signals that pass through the parameter estimates should satisfy a certain richness condition, such as having and maintaining full rank, throughout the training procedure. When gradient descent algorithm is used to train a neural network, it also creates an online learning problem. Consequently, robust estimation of the network parameters requires that both the training data and the hidden-layer activations fulfill a richness condition throughout the training procedure. In this work, we analyze the dynamics of the gradient descent algorithm on feedforward networks, and we derive sufficient conditions on the training data and the hidden-layer activations to guarantee boundedness of the model parameters after training. In light of the conditions derived, we propose a new training algorithm that improves the richness of the hidden-layer activations during training. We demonstrate that the proposed algorithm yields comparable margin characteristics on the training and test data for a network trained for a classification task with CIFAR-10 dataset.

1. Introduction

When a linear model is trained for a supervised learning task, the training data set needs to span the whole input space for the model parameters to be learned accurately. If this condition is not satisfied, the model will not be trained on a subspace in its domain, and the response of the model will be unpredictable for inputs containing any component in this subspace. If the model is trained by an iterative optimization algorithm, this richness condition on the training data set must be satisfied *throughout* the training procedure. Fulfilling such a requirement is in general not difficult for single-layer models with fixed input data, but it is a non-trivial problem when identifying unknown parameters of dynamical systems (Kumar & Varaiya, 1986; Sastry & Bodson, 1989), and as we will show in this work, when training multi-layer models.

Consider, for example, a linear dynamical system in $\mathbb{R}^n$:

$$h_{t+1} = Ah_t + Bx_t \quad \forall t \in \mathbb{N} \tag{1a}$$

$$y_t = Ch_t \quad \forall t \in \mathbb{N} \tag{1b}$$

with internal state $h_t \in \mathbb{R}^n$, input $x_t \in \mathbb{R}$, output $y_t \in \mathbb{R}$, and unknown parameters $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^n$ and $C \in \mathbb{R}^{1\times n}$. Assume we feed a constant input $\{x_t\}_{t\in\mathbb{N}}$ into this system, that is, we set $x_t = x_0$ for all $t \in \mathbb{N}$, and we try to learn the input-output relationship of the system by observing $\{y_t\}_{t\in\mathbb{N}}$ while the internal state of the system evolves. Note that as time increments, the eigenvalues of A with magnitude less than 1 cause the internal state h_t to lose its components in the eigenspaces corresponding to these eigenvalues. In other words, the stable eigenspaces of A vanish from $\{h_t\}_{t\in\mathbb{N}}$ exponentially fast, and consequently, not enough information is received about how the system behaves in these eigenspaces. As a result, the accurate input-output relationship of the system cannot be recovered.

This is a classical problem in system identification: when parameters of a dynamical system is estimated while the internal state of the system evolves, the input fed into the system needs to satisfy a certain richness condition. In particular, for successful identification of the system in (1), the input signal $\{x_t\}_{t\in\mathbb{N}}$ needs to contain a certain number of frequencies (Boyd & Sastry, 1983; 1986). As this example shows, training a dynamical model in an online fashion necessitates the injection of sustained perturbations into the system throughout the training procedure.

In this work, we show that similar requirements arise while training multi-layer models, even though there does not appear a dynamical model. As a preamble, consider a fixed data set $\{x_i\}_{i\in\mathcal{I}}$ and an L-layer feedforward neural network:

$$F(x) = f_L(f_{L-1}(\cdots(f_2(f_1(x))))),$$

where f_k represents the operation of the k-th layer of the network. The parameters of the k-th layer are excited by the

[1]University of California, Berkeley. Correspondence to: Kamil Nar <nar@eecs.berkeley.edu>.

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.

activations of the previous layer, $\{f_{k-1}(\cdots(f_1(x_i)))\}_{i\in\mathcal{I}}$. Therefore, robust estimation of these parameters rely on whether this set is rich enough and whether this richness is maintained throughout the training procedure.

1.1. Outline and Contributions

By analyzing the dynamics of the gradient descent algorithm on feedforward neural networks trained with the squared-error loss, we achieve the following.

1. For multi-layer networks with ReLU activations, we provide a richness condition on the hidden-layer activations that is sufficient for building a connection between the convergence of the gradient descent algorithm and the boundedness of the trained parameters. Note that this a prerequisite for the output of the network not to change arbitrarily when the input changes little. Then we show that this richness condition is not satisfied by a network trained naively with the gradient descent algorithm. This is expected given the existence of adversarial examples for naively trained models.

2. We reinterpret the classical regularization terms for single-layer linear models in terms of boosting the richness of the training data. By building an analogy, and in light of the sufficient richness conditions derived in this work, we introduce a training algorithm that improves the richness of the hidden-layer activations of multi-layer networks. Lastly, we demonstrate that this algorithm leads to similar margin characteristics on the training and test data when a network is trained for a classification task with CIFAR-10 data set.

1.2. Related Work

Implicit regularization of the gradient descent algorithm on matrix factorization, deep linear networks and multi-layer structures has recently been studied in (Arora et al., 2019; Gidel et al., 2019; Gunasekar et al., 2017; 2018; Du et al., 2018; Ji & Telgarsky, 2019). Our work also addresses the same subject, but the emphasis is on the necessary and sufficient richness requirements on the training data and hidden-layer activations for implicit regularization to take place. Furthermore, our analysis uses a non-vanishing learning rate, thereby elucidating the effect of learning rate on regularization.

Robustness of a model against small perturbations in its input is closely tied to presence of an effective regularization during its training. State-of-the-art neural networks, however, are known to lack this robustness; imperceptibly small perturbations can change their outputs drastically (Szegedy et al., 2013; Kurakin et al., 2016; Goodfellow et al., 2015). We demonstrate in this work that naively trained networks will fail to fulfill the richness requirements on the hidden-layer activations for implicit regularization, which provides an alternative understanding for the lack of robustness in these models.

We discuss other related subjects in Section 5, after presenting our results.

2. Richness of Hidden-Layer Activations

In this section, we consider a feedforward network with ReLU activations, which are denoted with the element-wise operation

$$(z)_+ = \begin{cases} z & z > 0, \\ 0 & z \leq 0. \end{cases}$$

The following theorem provides a sufficient richness condition for the hidden-layer activations of the network so that the convergence of the gradient descent algorithm implies the boundedness of the network parameters.

Theorem 1. *Consider an L-layer network with ReLU activations:*

$$\begin{aligned} &h_0(x) = x, \\ &h_j(x) = (W_j h_{j-1}(x))_+ \quad j = 1, 2, \cdots, L-1, \\ &h_L(x) = W_L h_{L-1}(x) \end{aligned}$$

with n_j nodes in its j-th hidden layer, and assume it has been trained by minimizing the squared-error loss with the gradient descent algorithm on the data set $\{x_i\}_{i\in\mathcal{I}}$. Let $\hat{W}_j$ and $\hat{h}_j$ denote the weight matrix and the output of the j-th layer after the training, and define $\mathcal{I}' = \{i \in \mathcal{I} : \hat{h}_L(x_i) \neq 0\}$. If all hidden-layer activations are bounded over the training data set:

$$\max_{i\in\mathcal{I}'} \max_{j\in[L]} \|\hat{h}_j(x_i)\|_2 < \infty,$$

and if every hidden-layer node is activated by a set of signals with full-rank in the preceding layer:

$$\sum\nolimits_{i\in\mathcal{I}'} \mathbb{I}\{e_k^\top \hat{W}_j \hat{h}_{j-1}(x_i) > 0\} \cdot \hat{h}_{j-1}(x_i)\hat{h}_{j-1}^\top(x_i) \succ \mathbf{0}$$

for all $k \in [n_j]$ and $j \in [L]$, then the convergence of the gradient descent algorithm implies the boundedness of $\hat{W}_j$ for all $j \in [L]$ for almost every initilization. □

Please refer to Appendix A for a detailed description of the bound on the trained parameters. Theorem 1 states that the convergence of the gradient descent algorithm guarantees the boundedness of the weight parameters if every hidden-layer node is activated by a set of activation patterns with a full rank in the preceding layer. This condition, however, is not easily satisfied, particularly by networks trained naively by the gradient descent algorithm.

To demonstrate this, we trained a 5-layer convolutional neural network for a binary classification task. The training

data was chosen as the two classes corresponding to the planes and the horses in CIFAR-10 data set. The network was trained with the squared-error loss. Figure 1 shows the principal component analysis of the signals in the hidden layers of the network after training. Note that the signals in the upper layers of the network can be explained by the first few principal components, which indicates that these signals are very low dimensional; and more importantly, the rank of these signals are lower than the width of corresponding layers. This shows that the richness condition described in Theorem 1 is not satisfied by this network. Note, however, that this is no surprise given that this naively-trained network is susceptible to adversarial examples; that is, minute changes in its input can change the output of the network drastically. In the next section, we will develop an alternative training method to produce richer sets of activations in the hidden-layers of the network.

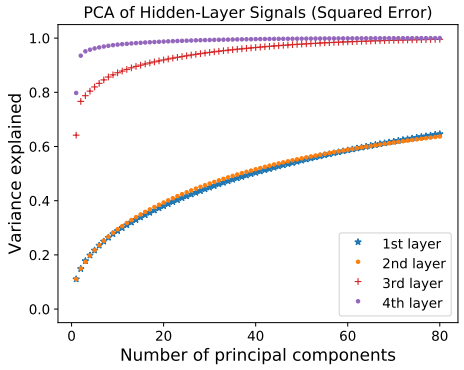

Figure 1. Principal component analysis for the activations in the hidden layers of a 5-layer convolutional neural network trained with the squared error loss. The plot displays the variance explained by the first 80 principal components for each layer. The layers have 4704, 1800, 120 and 84 nodes, respectively. The rank of the activation patterns in the upper layers are much lower than the number of nodes in those layers.

3. Reinterpreting Regularization

When a linear model is trained on a rank-deficient data set, or when a linear model is desired to be robust against small perturbations in its input, the classical procedure is introducing a regularization term to the training loss function. This regularization typically involves penalizing some norm of the model parameters, and it is considered to assign a prior distribution on the values the parameters can take or to limit the class of functions that can be learned.

We can provide a dual interpretation for these regularization terms in terms of the richness of the training data, as stated in the following theorem.

Theorem 2. *Given a set of points $\{x_i\}_{i\in\mathcal{I}}$ in $\mathbb{R}^n$ and corresponding target values $\{y_i\}_{i\in\mathcal{I}}$ in $\mathbb{R}$, consider the following two problems:*

$$\min_{w} \sum_{i\in\mathcal{I}} (y_i - w^\top x_i)^2 + \lambda \|w\|_p^m, \tag{2a}$$

$$\min_{w} \frac{1}{2} \sum_{i\in\mathcal{I}} \left[\Big(y_i - \min_{d:\|d\|_q\leq\epsilon} w^\top (x_i + d) \Big)^2 + \Big(y_i - \max_{d:\|d\|_q\leq\epsilon} w^\top (x_i + d) \Big)^2 \right], \tag{2b}$$

where $\|\cdot\|_p$ and $\|\cdot\|_q$ are dual norms, $m \in [1,\infty)$ is some fixed number, and $\lambda, \epsilon \in (0,\infty)$ are hyperparameters. For each λ, there exists some ϵ such that the solutions of the two problems are identical.

Problem (2b) shows that for each training point, perturbing the input of a linear mapping in two directions such that the output of the mapping is maximized and minimized creates a richness in the training data that is equivalent to penalizing the norm of the parameters. An analogous procedure for multi-layer neural networks is inserting perturbations to every hidden-layer of the network preceding the affine operations. To illustrate, consider an L-layer feedforward network with $h_0(x) = x$ and

$$h_j(x) = (W_j h_{j-1}(x) + b_j)_+ \quad \forall j \in [L], \tag{3}$$

where $\{W_j\}_{j\in[L]}$ and $\{b_j\}_{j\in[L]}$ are the weight and bias parameters of the network. The parameter W_j corresponds to a linear operation in the j-th layer of the network, such as a matrix multiplication or a convolution. We can insert perturbations to this network as $\tilde{h}_0(x;d) = x$ and

$$\tilde{h}_j(x;d) = (W_j[\tilde{h}_{j-1}(x;d) + d_j] + b_j)_+ \quad \forall j \in [L], \tag{4}$$

where $d = (d_1, d_2, \ldots, d_L)$ is the concatenation of the perturbations applied to each layer of the network. Then, solving the regression problem with the cost function

$$\sum_{i\in\mathcal{I}} \frac{1}{2}\Big(y_i - \min_{d\in\mathcal{D}} \tilde{h}_L(x_i;d) \Big)^2 + \frac{1}{2}\Big(y_i - \max_{d\in\mathcal{D}} \tilde{h}_L(x_i;d) \Big)^2,$$

where $\mathcal{D}$ is the allowed set of perturbations, should force the output of the network to remain close to the target values despite changes in the input and in the activations in the hidden layers.

Before proceeding to the next section, we summarize in Algorithm 1 the procedure for training a neural network while ensuring persistent excitation of the parameters. For simplicity, the algorithm is described for the stochastic gradient method with momentum.

4. Experimental Results

In this section, we test Algorithm 1 on a binary classification task. Only two classes of images, the horses and the planes,

Algorithm 1 Training with Persistent Excitation

input: training data $\{(x_i, y_i)\}_{i\in\mathcal{I}}$,
neural network $f_\theta(x; d) \equiv \tilde{h}_L(x; d)$ in (4),
set of allowed perturbations $\mathcal{D}$,
learning rate η, momentum parameter γ
initialize: $\Delta\theta \leftarrow 0$
repeat
  randomly choose $i \in \mathcal{I}$
  $d_1 \leftarrow \operatorname{argmax}_{d\in\mathcal{D}} f_\theta(x_i; d)$
  $d_2 \leftarrow \operatorname{argmin}_{d\in\mathcal{D}} f_\theta(x_i; d)$
  $\Delta\theta \leftarrow \gamma\Delta\theta + (1-\gamma)\nabla_\theta(f_\theta(x_i; d_1) - y_i)^2 + (1-\gamma)\nabla_\theta(f_\theta(x_i; d_2) - y_i)^2$
  $\theta \leftarrow \theta - \eta\Delta\theta$
until training is complete

have been chosen from CIFAR-10 data set for the classification task. The network used in the experiment contains two convolutional layers followed by three fully-connected layers.[1]

Figure 2 shows the margin distribution of the same network trained in two different ways: perturbing all layers of the network as described in Algorithm 1 and perturbing only the input of the network during training, similar to adversarial training (Madry et al., 2018). For both cases, the perturbations during the training phase are restricted to be in ℓ_∞ ball with radius 0.020. The plot shows the percentage of points the network misclassify versus the amount of disturbance needed in the input of the network to cause misclassification at evaluation phase, which is computed by using the projected gradient attack algorithm (Madry et al., 2018; Rauber et al., 2017) after training.

We observe that perturbing only the first layer of the network during training substantially increases the margin of the training data; however, this does not correspond to an improvement for the margin of the test data. In contrast, when all layers are perturbed as described in Algorithm 1 to improve the richness of the activations in the hidden layers, the margin of the training data becomes a good indicator of the margin of the test data.

5. Discussion

Dropout. Using dropout in the hidden-layer neurons, that is, setting the output of random subsets of neurons to zero during training, is known to prevent overfitting (Srivastava et al., 2014). This can be reinterpreted based on the richness of hidden-layer activations. Randomly setting some of the neurons in the hidden layers during training increases the variety in the hidden-layer activations, thereby improving

[1]The implementation is available at `github.com/nar-k/persistent-excitation`.

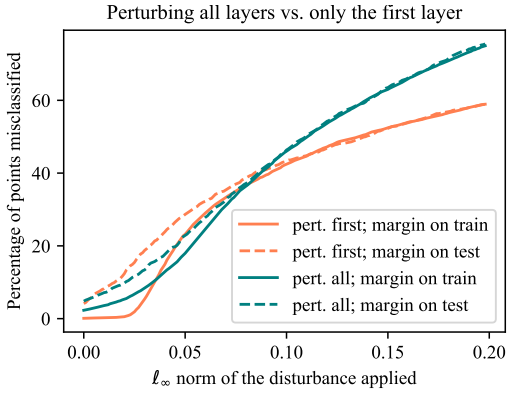

Figure 2. A network is trained in two different ways: by perturbing every layer of the network as described in Algorithm 1 in order to boost the richness of the hidden-layer activations, and by perturbing only the training data (the first layer) similar to adversarial training. When only the first layer is perturbed, the margin for the training data increases substantially, but this is not reflected in the test data. In contrast, when all layers are perturbed to improve the richness of the hidden-layer activations, the margin of the training data becomes a good indicator of the margin of the test data.

the richness of the signals that excite the parameters during training. Nevertheless, the space where the random perturbations is needed is very high-dimensional (it is equal to the dimension of the input space plus the total number of hidden-layer nodes in all layers of the network), and it is difficult to fill up the neighborhood of the training data in this space with random perturbations.

Importance of low-dimensional activations. The fact that activations in the hidden layers of a naively-trained neural network will become low-dimensional is critical in realizing that augmenting the training data set may not help achieving robustness in neural networks. This is because adding more training data will not necessarily be effective for attaining the required richness in the hidden layers, and consequently, the parameters may not be trained robustly. However, if the low-dimensionality of the hidden-layer activations is overlooked, the conclusion will be different. For example, (Schmidt et al., 2018) analyzes the robustness of classifiers with full-rank, non-degenerate data sets and single-layer, linear models, and arrives at the conclusion that the robustness could be achieved with more training data.

All-layer margin and generalization. (Wei & Ma, 2020) recently showed that a new concept of margin, which is computed for multi-layer models by allowing perturbations in the hidden-layers as well as in the inputs, can be used for obtaining generalization bounds for these models. This is closely aligned with our results involving the richness requirements on the hidden-layer activations of multi-layer models and the effect of boosting this richness by injecting perturbations into the hidden layers during training.

References

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 7411–7422, 2019.

Boyd, S. and Sastry, S. On parameter convergence in adaptive control. *Systems & Control Letters*, 3(6):311–319, 1983.

Boyd, S. and Sastry, S. Necessary and sufficient conditions for parameter convergence in adaptive control. *Automatica*, 22(6):629–639, 1986.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Du, S. S., Hu, W., and Lee, J. D. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In *Advances in Neural Information Processing Systems*, pp. 384–395, 2018.

Gidel, G., Bach, F., and Lacoste-Julien, S. Implicit regularization of discrete gradient dynamics in linear neural networks. In *Advances in Neural Information Processing Systems*, pp. 3196–3206, 2019.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Gunasekar, S., Woodworth, B. E., Bhojanapalli, S., Neyshabur, B., and Srebro, N. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 6151–6159, 2017.

Gunasekar, S., Lee, J. D., Soudry, D., and Srebro, N. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 9461–9471, 2018.

Ji, Z. and Telgarsky, M. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations*, 2019.

Kumar, P. R. and Varaiya, P. *Stochastic Systems: Estimation, Identification and Adaptive Control*. Prentice-Hall, Upper Saddle River, NJ, USA, 1986.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Rauber, J., Brendel, W., and Bethge, M. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

Sastry, S. *Nonlinear Systems: Analysis, Stability, and Control*. Springer-Verlag, 2013.

Sastry, S. and Bodson, M. *Adaptive Control: Stability, Convergence and Robustness*. Prentice Hall, 1989.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pp. 5014–5026, 2018.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

Wei, C. and Ma, T. Improved sample complexities for deep neural networks and robust classification via an all-layer margin. In *International Conference on Learning Representations*, 2020.

Appendices

A. Proof of Theorem 1

Theorem (Restatement of Theorem 1)**.** *Consider an L-layer network with ReLU activations:*

$$\begin{aligned} &h_0(x) = x, \\ &h_j(x) = (W_j h_{j-1}(x))_+ \quad j = 1, 2, \cdots, L-1, \\ &h_L(x) = W_L h_{L-1}(x) \end{aligned}$$

with n_j nodes in its j-th hidden layer, and assume it has been trained by minimizing the squared-error loss with the gradient descent algorithm on the data set $\{x_i\}_{i\in\mathcal{I}}$. Let $\hat{W}_j$ and $\hat{h}_j$ denote the weight matrix and the output of the j-th layer after the training, and define $\mathcal{I}' = \{i \in \mathcal{I} : \hat{h}_L(x_i) \neq 0\}$. Define $\mathcal{I}_j^k$ as the set of points that activate the k-th node in the j-th layer of the network after training:

$$\mathcal{I}_j^k = \{i \in \mathcal{I}' : e_k^\top \hat{h}_j(x_i) > 0\} \quad \forall k \in [n_j],\ \forall j \in [L-1].$$

Assume all hidden-layer activations are bounded over the training data set:

$$\max_{i\in\mathcal{I}'} \max_{j\in[L]} \|\hat{h}_j(x_i)\|_2 < \infty,$$

and every hidden-layer node is activated by a set of signals with full-rank in the preceding layer:

$$\sum\nolimits_{i\in\mathcal{I}_j^k} \hat{h}_{j-1}(x_i)\hat{h}_{j-1}^\top(x_i) \succ \mathbf{0} \quad \forall k \in [n_j], \forall j \in [L].$$

Then the convergence of the gradient descent algorithm from almost every initialization implies that

$$\|\hat{W}_j\|_2^2 \leq \frac{2}{\delta} \sum_{k\in[n_j]} \frac{1}{\lambda_{\min}(\sum_{i\in\mathcal{I}_j^k} \hat{h}_{j-1}(x_i)\hat{h}_{j-1}^\top(x_i))} \sum_{i\in\mathcal{I}_j^k} \frac{\|\hat{h}_j(x_i)\|_2^2}{\|\hat{h}_L(x_i)\|_2^2} \quad \forall j \in [L-1],$$

and

$$\|\hat{W}_L\|_2^2 \leq \frac{2}{\delta} \sum_{k\in[n_{L-1}]} \frac{1}{\max_{i\in\mathcal{I}_{L-1}^k} \|\hat{h}_{L-2}(x_i)\|_2^2}.$$

□

Proof. Given the neural network

$$\begin{aligned} &h_0(x) = x, \\ &h_j(x) = (W_j h_{j-1}(x))_+ \quad j = 1, 2, \ldots, L-1, \\ &h_L(x) = W_L h_{L-1}(x), \end{aligned}$$

for each point in $\{x_i\}_{i\in\mathcal{I}}$ and for each layer $j \in [L-1]$, define the diagonal activation matrix G_j^i as

$$(G_j^i)_{kk} = \begin{cases} 1 & \text{if } e_k^\top W_j h_{j-1}(x_i) > 0, \\ 0 & \text{otherwise,} \end{cases}$$

where e_k is the k-th standard basis vector in $\mathbb{R}^{n_j}$ for each $k \in [n_j]$. Then we can write

$$h_L(x_i) = W_L G_{L-1}^i W_{L-1} G_{L-2}^i W_{L-2} \cdots G_1^i W_1 x_i \quad \forall i \in \mathcal{I}.$$

The training loss function is

$$\frac{1}{2}\sum_{i\in\mathcal{I}}\|h_L(x_i)-y_i\|_2^2,$$

where $\{y_i\}_{i\in\mathcal{I}}$ is the set of target values corresponding to the input points. For each weight matrix W_j, we can write the gradient descent algorithm *calculated with automatic differentiation* as

$$\begin{aligned}W_j \leftarrow W_j - \delta\sum\nolimits_{i\in\mathcal{I}}\big(&G_j^iW_{j+1}^\top G_{j+1}^i\cdots W_L^\top W_L\cdots G_{j+1}^iW_{j+1}G_j^iW_jG_{j-1}^iW_{j-1}\cdots\\ &\cdots G_1^iW_1x_ix_i^\top W_1^\top G_1^i\cdots W_{j-1}^\top G_{j-1}^i\big)\\ +\delta\sum\nolimits_{i\in\mathcal{I}}\big(&G_j^iW_{j+1}^\top G_{j+1}^i\cdots W_L^\top y_ix_i^\top W_1^\top G_1^i\cdots W_{j-1}^\top G_{j-1}^i\big)\end{aligned}$$

followed by the updates of the activation matrices:

$$(G_j^i)_{kk}\leftarrow\mathbb{I}(W_jh_{j-1}(x_i)>0)\quad\forall k\in[n_j],\ \forall j\in[L-1],\ \forall i\in\mathcal{I}.$$

For the gradient descent algorithm to converge to an equilibrium from the points in its neighborhood, that equilibrium needs to be stable in the sense of Lyapunov (Sastry, 2013). Let $\{\hat{W}_j\}_{j\in[L]}$ and $\{\hat{G}_j^i\}_{i\in\mathcal{I},j\in[L-1]}$ denote the parameters and the activation matrices at equilibrium. Lyapunov stability of this equilibrium implies that

$$\lambda_{\max}\left(\hat{G}_j^i\hat{W}_{j+1}^\top\hat{G}_{j+1}^i\cdots\hat{W}_L^\top\hat{W}_L\cdots\hat{G}_{j+1}^i\hat{W}_{j+1}\hat{G}_j^i\right)\lambda_{\max}\left(\hat{h}_{j-1}(x_i)\hat{h}_{j-1}^\top(x_i)\right)<\frac{2}{\delta}.\tag{5}$$

Note that

$$\hat{W}_L\cdots\hat{G}_{j+1}^i\hat{W}_{j+1}\hat{G}_j^i\hat{h}_j(x_i)=\hat{h}_L(x_i),$$

and therefore,

$$\frac{\|\hat{h}_L(x_i)\|_2^2}{\|\hat{h}_j(x_i)\|_2^2}\le\lambda_{\max}\left(\hat{G}_j^i\hat{W}_{j+1}^\top\hat{G}_{j+1}^i\cdots\hat{W}_L^\top\hat{W}_L\cdots\hat{G}_{j+1}^i\hat{W}_{j+1}\hat{G}_j^i\right)\quad\forall i\in\mathcal{I}'.\tag{6}$$

Combining (5) and (6), we obtain

$$\lambda_{\max}\left(\hat{h}_{j-1}(x_i)\hat{h}_{j-1}^\top(x_i)\right)<\frac{2}{\delta}\frac{\|\hat{h}_j(x_i)\|_2^2}{\|\hat{h}_L(x_i)\|_2^2}\quad\forall i\in\mathcal{I}'.$$

On the other hand,

$$\lambda_{\max}\left(\hat{h}_{j-1}(x_i)\hat{h}_{j-1}^\top(x_i)\right)\ge e_k^\top\hat{G}_{j-1}^i\hat{W}_{j-1}\hat{h}_{j-2}(x_i)\hat{h}_{j-2}^\top(x_i)\hat{W}_{j-1}^\top\hat{G}_{j-1}^ie_k\quad\forall k\in[n_{j-1}].$$

Let $\mathcal{I}_j^k$ denote the set of points that activates the k-th node of the j-th layer at equilibrium:

$$\mathcal{I}_j^k=\{i\in\mathcal{I}':e_k^\top\hat{h}_j(x_i)>0\}\quad\forall k\in[n_j],\ \forall j\in[L-1].$$

Then we can write

$$\lambda_{\max}\left(\hat{h}_{j-1}(x_i)\hat{h}_{j-1}^\top(x_i)\right)\ge e_k^\top\hat{W}_{j-1}\hat{h}_{j-2}(x_i)\hat{h}_{j-2}^\top(x_i)\hat{W}_{j-1}^\top e_k\quad\forall i\in\mathcal{I}_{j-1}^k,$$

which implies

$$e_k^\top\hat{W}_{j-1}\hat{h}_{j-2}(x_i)\hat{h}_{j-2}^\top(x_i)\hat{W}_{j-1}^\top e_k\le\frac{2}{\delta}\frac{\|\hat{h}_j(x_i)\|_2^2}{\|\hat{h}_L(x_i)\|_2^2}\quad\forall i\in\mathcal{I}_{j-1}^k.$$

By summing over all points activating the k-th node of the $(j-1)$-th layer:

$$e_k^\top\hat{W}_{j-1}\left(\sum\nolimits_{i\in\mathcal{I}_{j-1}^k}\hat{h}_{j-2}(x_i)\hat{h}_{j-2}^\top(x_i)\right)\hat{W}_{j-1}^\top e_k\le\sum\nolimits_{i\in\mathcal{I}_{j-1}^k}\frac{2}{\delta}\frac{\|\hat{h}_j(x_i)\|_2^2}{\|\hat{h}_L(x_i)\|_2^2}.$$

This gives a bound on the k-th row of $\hat{W}_{j-1}$:

$$\|e_k^\top \hat{W}_{j-1}\|_2^2 \leq \frac{1}{\lambda_{\min}\left(\sum_{i\in\mathcal{I}_{j-1}^k} \hat{h}_{j-2}(x_i)\hat{h}_{j-2}^\top(x_i)\right)} \sum_{i\in\mathcal{I}_{j-1}^k} \frac{2}{\delta} \frac{\|\hat{h}_j(x_i)\|_2^2}{\|\hat{h}_L(x_i)\|_2^2}.$$

As a result, if

$$\sum_{i\in\mathcal{I}'} \mathbb{I}\{e_k^\top \hat{W}_{j-1}\hat{h}_{j-2}(x_i) > 0\} \cdot \hat{h}_{j-2}(x_i)\hat{h}_{j-2}^\top(x_i) \succ \mathbf{0} \quad \forall k \in [n_{j-1}],$$

all rows of $\hat{W}_{j-1}$ are bounded, for $j = 2, 3, \ldots, L$.

To bound the norm of $\hat{W}_L$, consider the gradient update for W_{L-1}. Similar to (5), we have

$$\lambda_{\max}\left(\hat{G}_{L-1}^i \hat{W}_L^\top \hat{W}_L \hat{G}_{L-1}^i\right) \lambda_{\max}\left(\hat{h}_{L-2}(x_i)\hat{h}_{L-2}^\top(x_i)\right) < \frac{2}{\delta} \quad \forall i \in \mathcal{I}'.$$

For every point activating the k-th node in the $(L-1)$-th layer, we have

$$\hat{G}_{L-1}^i e_k = e_k,$$

which yields

$$e_k^\top \hat{W}_L^\top \hat{W}_L e_k \leq \lambda_{\max}\left(\hat{G}_{L-1}^i \hat{W}_L^\top \hat{W}_L \hat{G}_{L-1}^i\right) \quad \forall i \in \mathcal{I}_{L-1}^k.$$

Therefore, we can write

$$\|\hat{W}_L e_k\|_2^2 \leq \frac{2}{\delta \|\hat{h}_{L-2}(x_i)\|_2^2} \quad \forall i \in \mathcal{I}_{L-1}^k,$$

which implies

$$\|\hat{W}_L e_k\|_2^2 \leq \frac{2}{\delta \max_{i\in\mathcal{I}_{L-1}^k} \|\hat{h}_{L-2}(x_i)\|_2^2}.$$

This proves that k-th column of $\hat{W}_L$ is bounded in norm. ■

B. Proof of Theorem 2

By duality of the norms $\|\cdot\|_p$ and $\|\cdot\|_q$, we have

$$\min_{d:\|d\|_q\leq\epsilon} w^\top(x_i + d) = w^\top x_i - \epsilon\|w\|_p,$$

$$\max_{d:\|d\|_q\leq\epsilon} w^\top(x_i + d) = w^\top x_i + \epsilon\|w\|_p.$$

Then problem (2b) can be written as

$$\min_w \sum_{i\in\mathcal{I}} \frac{1}{2}\left(y_i - w^\top x_i + \epsilon\|w\|_p\right)^2 + \frac{1}{2}\left(y_i - w^\top x_i - \epsilon\|w\|_p\right)^2,$$

which can be simplified to

$$\min_w \sum_{i\in\mathcal{I}} \left(y_i - w^\top x_i\right)^2 + \epsilon^2\|w\|_p^2.$$

This is a convex problem in w, and we can introduce a slack variable to bring the second term into a constraint form:

$$\begin{aligned} \underset{w,t}{\text{minimize}} \quad & \sum_{i\in\mathcal{I}} \left(y_i - w^\top x_i\right)^2 + \epsilon^2 t \\ \text{subject to} \quad & \|w\|_p^2 \leq t. \end{aligned} \tag{7}$$

Fix $\epsilon > 0$, and assume that (w_0, t_0) is the solution of (7). Then w_0 is also a solution to the problem

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \sum\nolimits_{i\in\mathcal{I}} (y_i - w^\top x_i)^2 \\ \text{subject to} \quad & \|w\|_p^2 \leq t_0, \end{aligned}$$

as well as

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \sum\nolimits_{i\in\mathcal{I}} (y_i - w^\top x_i)^2 \\ \text{subject to} \quad & \|w\|_p^m \leq t_0^{m/2}. \end{aligned} \tag{8}$$

If $t_0 = 0$, then w_0 is also zero, and this solution can be obtained by (2a) by choosing λ large enough. Therefore, without loss of generality assume $t_0 \neq 0$. Then problem (8) satisfies the Slater's condition, and strong duality holds (Boyd & Vandenberghe, 2004). Then we can find its solution by solving

$$\min_w \sum\nolimits_{i\in\mathcal{I}} (y_i - w^\top x_i)^2 + \lambda^* (\|w\|_p^m - t_0^{m/2})$$

where λ^* is the dual solution. Note that this problem is strictly convex in w, and therefore, its solution is unique, for which the only candidate is w_0. We conclude that

$$w_0 = \operatorname*{argmin}_w \sum\nolimits_{i\in\mathcal{I}} (y_i - w^\top x_i)^2 + \lambda^* \|w\|_p^m. \qquad \blacksquare$$