Persistency of Excitation for Robustness of Neural Networks

Kamil Nar NAR@EECS.BERKELEY.EDU
S. Shankar Sastry SASTRY@EECS.BERKELEY.EDU
Electrical Engineering and Computer Sciences, University of California, Berkeley

Abstract

When an online learning algorithm is used to estimate the unknown parameters of a model, the signals interacting with the parameter estimates should not decay too quickly for the optimal values to be discovered correctly. This requirement is referred to as persistency of excitation, and it arises in various contexts, such as optimization with stochastic gradient methods, exploration for multi-armed bandits, and adaptive control of dynamical systems. While training a neural network, the iterative optimization algorithm involved also creates an online learning problem, and consequently, correct estimation of the optimal parameters requires persistent excitation of the network weights. In this work, we analyze the dynamics of the gradient descent algorithm while training a two-layer neural network with two different loss functions, the squared-error loss and the cross-entropy loss; and we obtain conditions to guarantee persistent excitation of the network weights. We then show that these conditions are difficult to satisfy when a multi-layer network is trained for a classification task, for the signals in the intermediate layers of the network become low-dimensional during training and fail to remain persistently exciting. To provide a remedy, we delve into the classical regularization terms used for linear models, reinterpret them as a means to ensure persistent excitation of the model parameters, and propose an algorithm for neural networks by building an analogy. The results in this work shed some light on why adversarial examples have become a challenging problem for neural networks, why merely augmenting training data sets will not be an effective approach to address them, and why there may not exist a data-independent regularization term for neural networks, which involve only the model parameters but not the training data.

Keywords: Neural networks, Robustness of neural networks, Regularization for neural networks, Persistency of excitation, Adversarial examples.

The implementation of all experiments is available at https://github.com/nar-k/persistent-excitation.

Contents

1. Introduction

State-of-the-art neural networks provide high accuracy for plenty of machine learning tasks, but their performance has proven vulnerable to small perturbations in their inputs (Szegedy et al., 2013). This lack of robustness prohibits the use of neural networks in safety-critical applications such as computer security, control of cyber-physical systems, self-driving cars, and medical prediction and planning (Kurakin et al., 2016; Strickland, 2019).

Given the combination of large number of training parameters, nonlinearity of the model, nonconvexity of the optimization problem used for training, involvement of an iterative algorithm for optimization, and high-dimensionality of the commonly used data sets, it is challenging to pinpoint what particularly causes the lack of robustness in neural networks. It has been speculated that the high nonlinearity of neural networks gives rise to this vulnerability (Szegedy et al., 2013), but the fact that support vector machines with nonlinear feature mappings remain much more robust against similar perturbations readily invalidates this claim (Goodfellow et al., 2015).

It has been observed that the lack of robustness is not a consequence of the depth of the network; even networks with few layers are shown to produce drastically different outputs for almost identical inputs. Comparison of this observation with the robustness of support vector machines has led to the belief that neural networks might in fact not be introducing sufficient level of nonlinearity to execute the learning tasks involved — since shallow networks function as an affine mapping in most of their domains, whereas most feature mappings used in support vector machines are unarguably nonlinear (Goodfellow et al., 2015). Although this belief still prevails, the level of nonlinearity is not the only aspect that neural networks and support vector machines differ in. There are many other factors these two structures do not share in common, and these factors might also account for their dissimilarity in robustness.

One of the important yet overlooked differences between neural networks and support vector machines is the first-line loss functions used in their training. For classification tasks, neural networks are almost always trained with the cross-entropy loss, whereas training support vector machines involves the hinge loss function (Hastie et al., 2009). This raises the question whether the choice of training loss function, and in particular the cross-entropy loss function, is one of the factors leading to the lack of robustness in neural networks. To provide a preliminary answer to this question, we train a two-layer neural network for a binary classification task with two different loss functions: the cross-entropy loss and the squared-error loss. Figure 1 demonstrates the decision boundaries of the network trained with the two loss functions for different initializations. Even though the network architecture, the training data, and the optimization algorithm are kept identical, Figure 1 shows that the network trained with the cross-entropy loss consistently has a much smaller margin than the network trained with the squared-error loss, and hence, it is less robust.

Figure 1 confirms that at least some ingredients of neural network training procedure have an influence on the robustness of networks. Naturally, the next question to ask is whether the identical training procedure could lead to a similar vulnerability in models that are different than neural networks. Figure 2 illustrates two examples of a linear classifier trained with the cross-entropy loss function and the gradient descent algorithm. It appears that the decision boundary of a linear classifier could also have an extremely poor margin when it is trained in exactly the same way as neural networks are trained.

To understand the observations in Figure 1 and Figure 2, we will first look into a concept called persistency of excitation, and then explain how it relates to training neural networks.

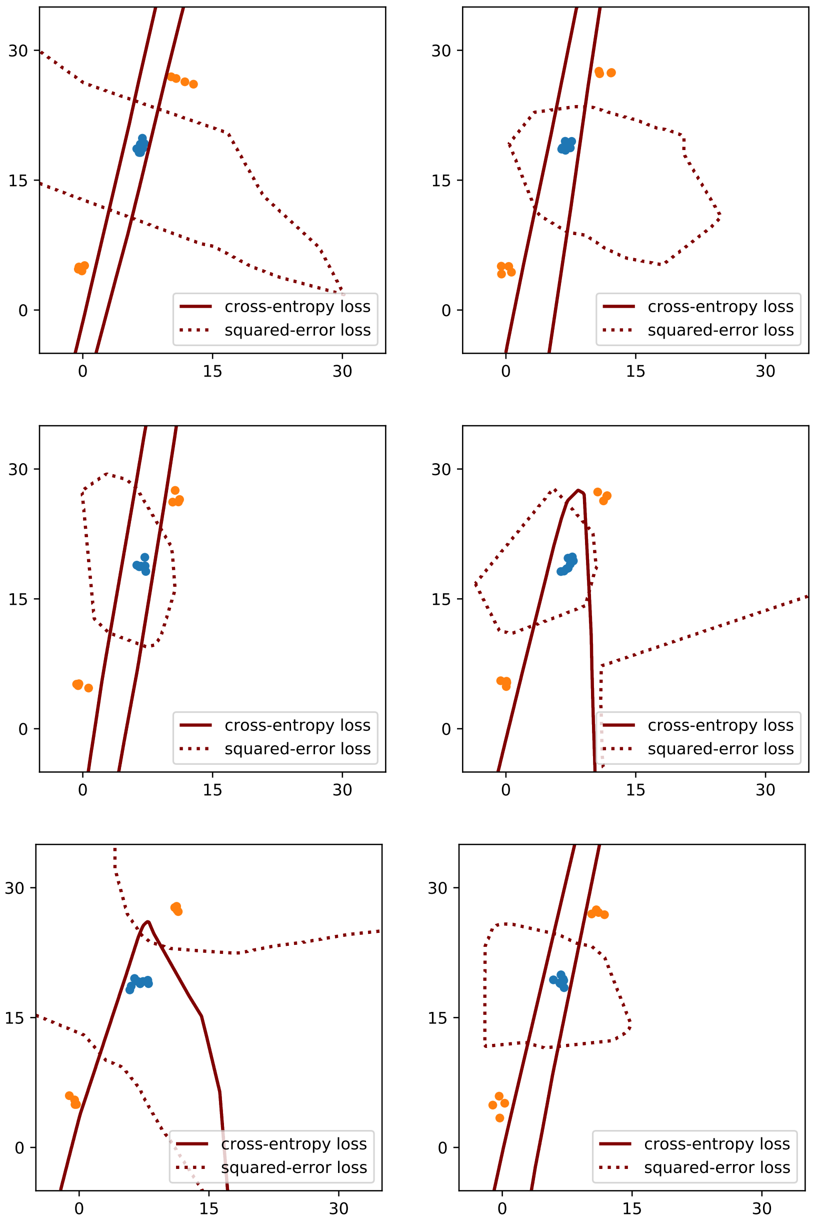

Figure 1: The decision boundaries of a two-layer network trained with the cross-entropy loss and the squared-error loss. The task is binary classification in $\mathbb{R}^2$; the orange clusters form one class and the blue cluster represents the other class. Each plot shows the decision boundaries for a different initialization, corresponding to the random seeds 1, 21, 41, 61, 81, 101. The network architecture, the optimization algorithm, and the training data are identical within each plot; only the training loss functions are different. The networks trained with the cross-entropy loss consistently have a substantially poorer margin.

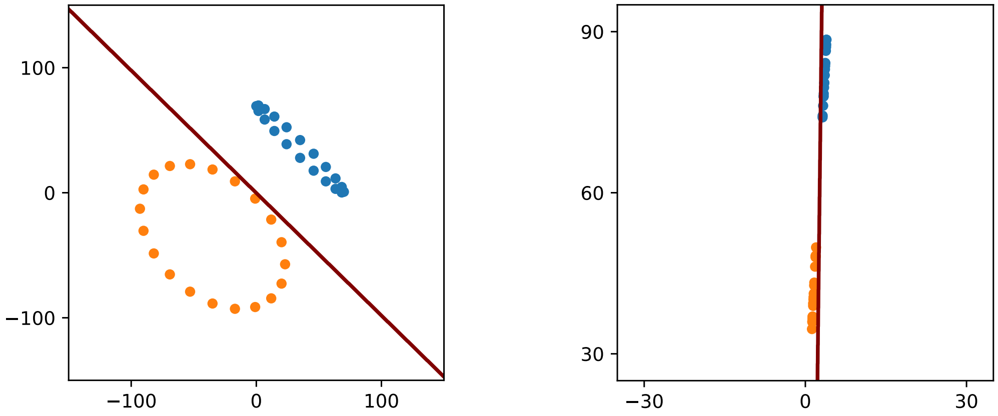

Figure 2: Two different data sets on which a linear classifier is trained by minimizing the cross-entropy loss using the gradient descent algorithm. The solid lines represent the decision boundary of the classifier, which lie very close to the training data. Standard ingredients of neural network training seem to cause extremely poor margin and lack of robustness for linear classifiers as well.

1.1. Persistency of Excitation

Consider the training of a linear model with the squared-error loss:

$$\min_{\theta} \sum\nolimits_{i \in \mathcal{I}} \|x_i^\top \theta - y_i\|_2^2,$$

where $\{x_i\}_{i \in \mathcal{I}}$ is the set of training data in $\mathbb{R}^n$ and $\{y_i\}_{i \in \mathcal{I}}$ is the set of target values in $\mathbb{R}$. This problem is convex, and if the set $\{x_i\}_{i \in \mathcal{I}}$ spans whole $\mathbb{R}^n$, the optimal value for θ will be unique. We can use the gradient descent algorithm to find this optimal value:

$$\theta \leftarrow \theta - \delta \sum\nolimits_{i \in \mathcal{I}} x_i (x_i^\top \theta - y_i), \tag{1}$$

where δ is some fixed step size. As long as δ is sufficiently small, the parameter estimate is guaranteed to converge to the optimal value of θ.

Now, assume for some reason the measurements are attenuated at each iteration of the gradient descent algorithm. That is, the update rule (1) is followed at each iteration by

$$(x_i, y_i) \leftarrow (\alpha x_i, \alpha y_i) \quad \forall i \in \mathcal{I},$$

for some $\alpha \in (0, 1)$. In this case, the gradient descent algorithm will lose its ability to find the true value of θ. For every initialization, the algorithm will still converge, the error term will still become zero, but the parameters will not necessarily converge to the optimal θ.

This might seem like a contrived example, but it is a fundamental problem that arises in various settings, such as identification and adaptive control of dynamical systems, optimization with stochastic gradient methods, and exploration in multi-armed bandits.

- **System identification and adaptive control:** Consider a discrete-time linear dynamical system:
$$x_{t+1} \leftarrow \theta x_t + u_t \quad \forall t \in \mathbb{N}, \tag{2}$$
where $x_t \in \mathbb{R}$ and $u_t \in \mathbb{R}$ are the state and the input of the system at time t, and $\theta \in (0, 1)$ is the unknown parameter of the system. Assume an estimator system of the form
$$\hat{x}_{t+1} \leftarrow \hat{\theta}\hat{x}_t + u_t \quad \forall t \in \mathbb{N}$$
is used to identify the value of θ by decreasing the distance between x_t and $\hat{x}_t$ over time with the gradient descent algorithm. Then both x_t and $\hat{x}_t$ can decay to zero while $\hat{\theta}$ converges to some value different than θ (Kumar and Varaiya, 1986).

- **Stochastic gradient methods:** Consider a convex function $f(\theta)$, which can be decomposed as $\sum_{i\in\mathcal{I}} f_i(\theta)$, where each f_i is also a convex function. Assume we use the following stochastic gradient method with a fixed step size δ:

  1. Randomly choose $i \in \mathcal{I}$,
  2. $\theta_{t+1} \leftarrow \theta_t - \delta \frac{\partial}{\partial\theta} f_i(\theta)$,
  3. Scale down $f(\theta)$; in other words, set $f_i \leftarrow \alpha_t f_i$ for all $i \in \mathcal{I}$ for some $\alpha_t \in (0, 1)$,
  4. Return to 1.

  This algorithm does not necessarily converge to the optimal value for θ if α_t attenuates $f(\theta)$ too quickly (Bottou et al., 2018).

- **Multi-armed bandits:** Consider a two-armed bandit problem where the arms produce the independent and identically distributed random processes $\{X_t\}_{t\in\mathbb{N}}$ and $\{\tilde{X}_t\}_{t\in\mathbb{N}}$ with distinct means; that is, $\mathbb{E}X_t \neq \mathbb{E}\tilde{X}_t$. Assume we try to solve
$$\max_{\{d_t\}_{t\in\mathbb{N}}\in\{0,1\}^{\mathbb{N}}} \mathbb{E}\left(\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\left(d_t X_t + (1-d_t)\tilde{X}_t\right)\right)$$
by using a causal feedback mechanism $f : \mathbb{N}^2 \times \mathbb{R}^2 \mapsto \{0, 1\}$ that prescribes a decision at time $t \in \mathbb{N}$ as:
$$d_t = f\left(t, \sum\nolimits_{t'=1}^{t-1} d_{t'}, \sum\nolimits_{t'=1}^{t-1} d_{t'}X_{t'}, \sum\nolimits_{t'=1}^{t-1}(1-d_{t'})\tilde{X}_{t'}\right).$$
If the feedback policy f decreases the frequency of pulling the arm with the lower average of observed rewards too quickly, then the algorithm can fail to discover the arm with the higher mean (Bubeck and Cesa-Bianchi, 2012).

In these three examples, there is an input, a signal, or an action that excites the unknown parameters; that is, something that forces the parameters to release some information about themselves. We observe that when this excitation decays too quickly, the online learning algorithm cannot receive necessary amount of information about the parameters and fail to estimate them correctly. This leads us to a concept called *persistency of excitation*. For online algorithms to learn the unknown parameters of a model correctly, the signals interacting with the parameter estimates need to remain persistently exciting during the estimation procedure (Kumar and Varaiya, 1986; Sastry and Bodson, 1989). If this persistency of excitation condition is not satisfied, the error terms inside the learning algorithm can become zero even if the parameters converge to a wrong value or do not converge at all.

1.2. Connection Between Persistency of Excitation and Neural Networks

During training of a feedforward neural network, the iterative optimization algorithm involved creates the dynamics of an online learning problem as shown in Figure 3. The function f_θ represents the true mapping with the ideal parameters, whereas $f_{\hat{\theta}}$ denotes the estimate obtained with the gradient descent algorithm. Assume that the neural network has L layers, and it is described as

$$f_\theta(x) = f_{\theta_L} \circ f_{\theta_{L-1}} \circ \cdots \circ f_{\theta_1}(x),$$

where f_{θ_k} represents the operation the k-th layer of the network performs and θ_k stands for the parameters of that particular layer for each $k \in [L]$. Given a set of training data $\{x_i\}_{i\in\mathcal{I}}$, the parameters of the k-th layer, θ_k, are excited by the signals of the preceding layer throughout the training, which are given as $\{f_{\theta_{k-1}} \circ \cdots \circ f_{\theta_1}(x_i)\}_{i\in\mathcal{I}}$. Note that no matter how large the training data set is, there is no strong reason to think that these signals in the intermediate layers will remain persistently exciting during training, and we will show in this work that they indeed do not remain persistently exciting with standard training procedures. The consequence is that even if the training error converges to zero, the estimated parameters may not be the same as the ideal parameters of the true mapping, and they may not predict the output of the network accurately for unseen data.

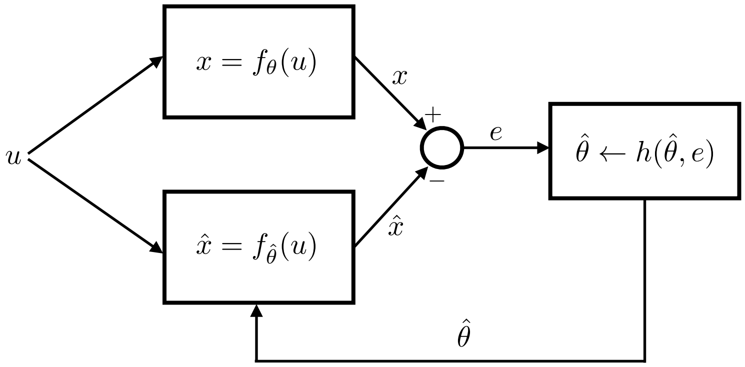
$x = f_\theta(u)$
x
$+$
e
$\hat{\theta} \leftarrow h(\hat{\theta}, e)$
u
$-$
$\hat{x}$
$\hat{x} = f_{\hat{\theta}}(u)$
$\hat{\theta}$

Figure 3: Closed-loop representation of the dynamics of neural network training. Function f_θ represents the true mapping that we want to estimate, while $f_{\hat{\theta}}$ represents the mapping estimated via the gradient descent algorithm. Signal e denotes the training error of the model estimated, and function h gives the update rule for the parameter estimates.

We will see in the following sections that there exist some conditions on the training data which ensure the persistent excitation of the network parameters. Given these conditions, convergence of training error to zero will imply the convergence of the estimate $\hat{\theta}$ to an optimal parameter θ that will induce the network to produce similar outputs for similar inputs.

The concept of persistency of excitation already provides an interpretation of Figure 1 and Figure 2. Different training loss functions give rise to different dynamics for the gradient descent algorithm; hence, they involve different conditions on the training data for the persistent excitation of the parameters — some of which are easier to satisfy than the others. Moreover, for some loss functions, even the linear models are not exempted from the restrictiveness of these conditions.

1.3. Our Contributions and Outline

The next section starts with the analysis of the gradient descent algorithm on the squared-error loss function. It provides a richness condition on the training data set such that the convergence of the gradient descent algorithm guarantees a small Lipschitz constant for the estimated model.

Section 3 studies the dynamics of the gradient descent algorithm on the cross-entropy loss function. For linear and nonlinear classifiers, conditions for persistency of excitation are derived and shown to be different than the conditions required for the squared-error loss. This difference will clarify the contrast between the decision boundaries of the two loss functions in Figure 1.

It has been consistently observed in recent works (Martin and Mahoney, 2019) that neural networks generate low-dimensional signals in their hidden layers. Section 4 provides a simple theoretical analysis confirming this phenomenon for deep linear networks. This fact suggests the conditions given in Section 2 and Section 3 for persistency of excitation are in fact difficult to satisfy on multiple-layer neural networks trained with the gradient descent algorithm.

To address the problem raised in Section 4, we will look closely into the classical approach that is used to strengthen the correspondence between the training and test performances of a model: adding regularization. The traditional viewpoint is that adding a regularization term to the training loss function assigns a prior distribution to the parameters of the model and restricts the class of functions that can be represented by the model (Hastie et al., 2009). Section 5 introduces a dual viewpoint: the regularization term creates a richer training data set and eases meeting the condition for the persistency of excitation. By building an analogy with this viewpoint, a new training scheme is introduced to ensure persistent excitation of all parameters of a neural network during its training.

Section 6 contains the experimental results of the algorithm proposed for persistent excitation of neural networks. Section 7 discusses the results and provides a comparison with the related works, and finally Section 8 concludes the paper. Appendices contain the proofs of the theorems and the implementation details for the experiments.

2. Gradient Descent on Squared-Error Loss

Given a regression problem with the squared-error loss, it is natural to expect that the Lipschitz constant of the estimate will depend on the Lipschitz constant of the actual function generating the target values for the input points. Therefore, one might anticipate determining the Lipschitz constant of the estimate will require information about the relationship between the input points and their target values. This indeed is the case for linear estimators, but not for neural networks. The following subsections will show that when a neural network is trained with the gradient descent algorithm, the Lipschitz constant of the estimate will have a bound depending only on the set of training points but *not* on their target values. In other words, the Lipschitz constant of the estimate will be restricted no matter how sharply the actual function changes in its domain.

2.1. Lack of Input-Based Lipschitz Bounds for Linear Estimators

Consider again the problem of finding a linear estimator for a function by minimizing its squared-error loss over a set of points $\{x_i\}_{i\in\mathcal{I}}$. Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be the function that is being estimated, and let $\ell : \mathbb{R}^{m\times n} \to \mathbb{R}$ denote the loss function that is being minimized:

$$\ell(W) = \frac{1}{2}\sum\nolimits_{i\in\mathcal{I}} \|Wx_i - f(x_i)\|_2^2 \,.$$

If we use the gradient descent algorithm with a fixed step size δ to minimize ℓ, the update rule for W becomes

$$W \leftarrow W\left(I - \delta \sum\nolimits_{i\in\mathcal{I}} x_i x_i^\top\right) + \delta \sum\nolimits_{i\in\mathcal{I}} f(x_i) x_i^\top,$$

which is a discrete-time linear time invariant system. Therefore, a necessary and sufficient condition for the convergence of the algorithm is that (Callier and Desoer, 2012)

$$\lambda_{\max}\left(\sum\nolimits_{i\in\mathcal{I}} x_i x_i^\top\right) < \frac{2}{\delta}. \tag{3}$$

The left hand side of (3) is the largest eigenvalue of the covariance matrix of the training data, which is nothing but the Lipschitz constant of the gradient of the loss function ℓ.

This result is well-known (Bertsekas, 1999): when a differentiable function $\ell : \mathbb{R}^n \to \mathbb{R}$ is minimized by using the gradient descent algorithm with a fixed step size δ, the algorithm can converge to a local minimum $w_0 \in \mathbb{R}^n$ only if the Lipschitz constant of the gradient of ℓ, $L(\nabla\ell)$, at that point satisfies

$$L\left(\nabla\ell(w_0)\right) < \frac{2}{\delta}.$$

This condition for convergence establishes a relationship between the smoothness parameter of the loss function and the step size of the algorithm.

The inequality (3), however, does *not* have any implication on the Lipschitz constant of *the estimate for f* — which is the largest singular value of $\hat{W}$. Indeed, the spectral radius of $\hat{W}$ could be arbitrarily large; it does not require the step size to be small for convergence. In contrast, Theorem 1 in the following subsection shows that when a neural network is being trained with the gradient descent algorithm, the step size does give information about the Lipschitz constant of the function estimate. This forms the crucial difference between the result of Theorem 1 and the well-known relationship between the step size and *some* Lipschitz constant.

2.2. Input-Based Lipschitz Bounds for Two-Layer Neural Networks

Now consider a two-layer neural network with ReLU activation in its hidden layer:

$$x \mapsto W(Vx + b)_+$$

where $W \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{r\times n}$ and $b \in \mathbb{R}^r$. The following theorem shows that if this network is trained to estimate a function $f : \mathbb{R}^n \to \mathbb{R}^m$ by minimizing the squared error loss via gradient descent algorithm, then the convergence of the algorithm implies Lipschitzness of the function estimate $\hat{f}(x) = \hat{W}(\hat{V}x + \hat{b})_+$ as long as the training data set is rich enough.

Theorem 1 *Given a function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a set of points $\{x_i\}_{i\in\mathcal{I}}$ in $\mathbb{R}^n$, assume that a two-layer neural network with parameters $W \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{r\times n}$, $b \in \mathbb{R}^r$ and ReLU activations is trained by minimizing the squared error loss*

$$\min_{W,V,b} \frac{1}{2} \sum\nolimits_{i\in\mathcal{I}} \|W(Vx_i + b)_+ - f(x_i)\|_2^2 \tag{4}$$

via the gradient descent algorithm. Assume that the algorithm has converged from a random initialization[1] *to a stationary point $(\hat{W}, \hat{V}, \hat{b})$ with a fixed step size δ. Let $\hat{V}_k$ and $\hat{b}_k$ denote the k-th*

[1] For this theorem, and all other theorems in this paper, the distribution used for random initialization is assumed to assign zero probability to every set of Lebesgue measure zero.

rows of $\hat{V}$ and $\hat{b}$ at equilibrium, and define the set of training points that activate the k-th node in the hidden layer as

$$\mathcal{I}_k = \big\{ i \in \mathcal{I} : \hat{V}_k x_i + \hat{b}_k > 0 \big\}.$$

Then the Lipschitz constant of the function estimate $\hat{f}(x) = \hat{W}(\hat{V}x + \hat{b})_+$ is almost-surely upper bounded by

$$\mathcal{L}(\delta, \overline{\mu}, \underline{\lambda}, \hat{b}, n_{\text{active}}^{\max}) := n_{\text{active}}^{\max} \left(\sqrt{\frac{2}{\delta \underline{\lambda}}} \right) \left(\frac{2\overline{\mu} \|\hat{b}\|_\infty}{\underline{\lambda}} + \sqrt{\frac{1}{\underline{\lambda}} \left| \frac{2}{\delta} - \|\hat{b}\|_\infty^2 \right|} \right) \tag{5}$$

where

- *$n_{\text{active}}^{\max}$ is the maximum number of hidden-layer nodes that a point in $\mathbb{R}^n$ can activate:*

$$n_{\text{active}}^{\max} = \max_{x \in \mathbb{R}^n} \sum\nolimits_{k=1}^{r} \mathbf{1}_{\{\hat{V}_k x + \hat{b}_k > 0\}},$$

- *$\underline{\lambda}$ is a lower bound for the minimum-eigenvalue of the covariance matrix of the training points that activate the same hidden-layer node:*

$$\underline{\lambda} = \min_{k \in [r]} \lambda_{\text{min}} \Big(\sum\nolimits_{i \in \mathcal{I}_k} x_i x_i^\top \Big),$$

- *$\overline{\mu}$ is an upper bound for the ℓ_2 norm of the summation of the training points that activate the same hidden-layer node:*

$$\overline{\mu} = \max_{k \in [r]} \Big\| \sum\nolimits_{i \in \mathcal{I}_k} x_i \Big\|_2.$$

Proof See Appendix A. ■

Theorem 1 underlines a fact: convergence of the training algorithm guarantees a Lipschitz bound for the network, from its input to its output, provided that every node in the hidden layer is excited from every dimension, i.e., the set of points that activate each hidden layer node spans the whole input space. This provides a sufficient condition for the Lipschitzness of the network in terms of the richness of the training data.

Theorem 1 also reveals the following:

1. Convergence with a larger step size implies a smaller Lipschitz constant for the mapping represented by the network, from its input to its output. This extends the similar result shown for deep linear networks in (Nar and Sastry, 2018) to two-layer nonlinear networks.

2. If the set of points activating any one of the hidden layer nodes has a large bias, then $\overline{\mu}$ will be large, and so will the upper bound for the Lipschitz constant.

3. As the number of hidden-layer nodes increases, so does the bound for the Lipschitz constant. This might seem to contradict with the premise of (Bartlett, 1998), which establishes that the weights of the network parameters are more important than the size of the network for its performance on training and test data sets to be similar. However, Bartlett (1998) studies a network with its parameters specified a priori — independently of whether those weights could be obtained by an iterative algorithm or not. Theorem 1, on the other hand, shows that if the size of the network is large, the gradient descent algorithm can potentially drive the network weights in such a way that the Lipschitz constant of the network becomes large.

If a network is trained for a classification task, the small Lipschitz constant of the network, from its input to its output, will suggest a large margin between the training data and the decision boundary of the classifier. This is formalized in the following corollary.

Corollary 1 *Given two sets of points $\{x_i\}_{i\in\mathcal{I}}$ and $\{x_j\}_{j\in\mathcal{J}}$ in $\mathbb{R}^n$, assume that a two-layer neural network is trained to classify them by minimizing the squared-error loss:*

$$\min_{W,V,b} \frac{1}{2}\sum\nolimits_{i\in\mathcal{I}} \|W(Vx_i+b)_+ - 0\|_2^2 + \frac{1}{2}\sum\nolimits_{j\in\mathcal{J}} \|W(Vx_j+b)_+ - 1\|_2^2$$

via the gradient descent algorihtm. If the algorithm converges from a random initialization to a solution $\hat{f}(x) = \hat{W}(\hat{V}x+\hat{b})_+$ with a step size δ, and if

$$\Delta := \min\nolimits_{j\in\mathcal{J}} \hat{f}(x_j) - \max\nolimits_{i\in\mathcal{I}} \hat{f}(x_i) > 0,$$

then the classifier

$$x \mapsto \text{sign}\left(\hat{f}(x) - \max\nolimits_{i\in\mathcal{I}} \hat{f}(x_i) - \Delta/2\right)$$

has a margin (in the input space) greater than or equal to

$$\frac{\Delta}{2\mathcal{L}(\delta, \overline{\mu}, \underline{\lambda}, b, n_{active}^{max})}$$

almost surely, where $\mathcal{L}$ is as defined in (5). ■

The same relationship between a small Lipschitz constant and a large margin will hold for the similar results in the following sections.

3. Gradient Descent on Cross-Entropy Loss

Logistic regression and its counterpart for multiple classes, cross-entropy minimization with the soft-max function, are widely used for classification tasks. In this section, we analyze the dynamics of the gradient descent algorithm on these loss functions and show how the concept of persistent excitation arises if a guarantee for a large margin is needed for the classifier.

Note that state-of-the-art neural networks are observed to achieve zero training error for classification tasks (Zhang et al., 2017), which implies that the features of the training points eventually become linearly separable in some layer of the networks during the training procedure. For this reason, we focus our analysis on the case where the features of the training data points are linearly separable.

3.1. Margin Analysis for Linear Classifier

When a linear classifier is trained with logistic regression for a binary classification task, the optimization problem involved is known to be convex. Nevertheless, this optimization problem cannot attain its minimum if the training points are linearly separable, that is, if there is a hyperplane separating the points of one class from the others (Hastie et al., 2009). The objective function could be made arbitrarily close to zero by driving the parameters to infinity in norm. Then the direction of the growth of the parameters becomes crucial, since this direction determines the normal vector of the separating hyperplane, and consequently, the margin of the classifier.

In recent work of Soudry et al. (2018), it has been shown that when the training data set is linearly separable, the separating hyperplane obtained with cross-entropy minimization via the gradient descent algorithm is described by a support vector machine problem in an augmented space — with the caveat that the hyperplane is constrained to pass through the origin in this augmented space. For this reason, the margin obtained with the cross-entropy minimization is suboptimal in the original space the training points lie in. The following theorem demonstrates how this margin compares to the true optimal margin.

Theorem 2 *Assume that the points $\{x_i\}_{i\in\mathcal{I}}$ and $\{x_j\}_{j\in\mathcal{J}}$ are linearly separable, and a linear classifier is trained by minimizing the cross entropy loss:*

$$\min_{w,b} \sum\nolimits_{i\in\mathcal{I}} \log\left(1+e^{-w^\top x_i - b}\right) + \sum\nolimits_{j\in\mathcal{J}} \log\left(1+e^{w^\top x_j + b}\right),$$

via the gradient descent algorithm. Let $\langle \overline{w}, \cdot\rangle + B = 0$ denote the decision boundary obtained, and assume that $\overline{w}$ and B are scaled such that

$$\min_{i\in\mathcal{I}} \langle \overline{w}, x_i\rangle - \max_{j\in\mathcal{J}} \langle \overline{w}, x_j\rangle = 2.$$

Define the set of indices for the support vectors as

$$\mathcal{I}_{sup} = \left\{ i\in\mathcal{I} : \langle \overline{w}, x_i\rangle \le \langle \overline{w}, x_{i'}\rangle \ \forall i'\in\mathcal{I}\right\},$$
$$\mathcal{J}_{sup} = \left\{ j\in\mathcal{J} : \langle \overline{w}, x_j\rangle \ge \langle \overline{w}, x_{j'}\rangle \ \forall j'\in\mathcal{J}\right\}.$$

If the support vectors lie in an affine subspace, that is, if there exist a set of orthonormal vectors $\{r_k\}_{k\in K}$ and a set of scalars $\{\Delta_k\}_{k\in K}$ such that

$$\langle r_k, x_i\rangle = \langle r_k, x_j\rangle = \Delta_k \quad \forall i\in\mathcal{I}_{sup},\ \forall j\in\mathcal{J}_{sup},\ \forall k\in K,$$

then the minimization of the cross-entropy loss yields a margin smaller than or equal to

$$\frac{1}{\sqrt{\frac{1}{\gamma_{OPT}^2} + B^2 \sum_{k\in K} \Delta_k^2}}$$

where γ_{OPT} denotes the optimal hard margin in the input space given by the SVM solution.

Proof See Appendix B. ■

Theorem 2 points out that the margin of the classifier is determined by only the support vectors, which are the training points that are closest to the decision boundary of the classifier. The points that are further to the decision boundary have no effect on the margin, and there is a reason for this: their *excitation* of the parameters do not *persist* as the gradient descent algorithm continues to update the parameters. Once these points are correctly classified, their contribution to the gradient of the loss function is removed exponentially fast, and consequently, the dynamics of the algorithm is dominated by the other points.

Theorem 2 also shows that the margin obtained with cross-entropy minimization will be much smaller than the optimal value if the support vectors of the data set lie in a low dimensional affine subspace. Furthermore, as the dimension of this subspace decreases, the set K in Theorem 2 grows in cardinality, and the margin of the classifier gets even smaller.

It is essential to note that the bias term B in Theorem 2 is *not* the bias of the training data set; it is the bias of the support vectors. It is likely that the training data points have zero mean, whereas the support vectors have non-negligible bias. In fact, the plot on the left of Figure 2 is intentionally chosen for this reason; it contains a data set with zero mean.

Note also that making the bias term B zero requires the a priori knowledge of the set of support vectors — which is not available until the algorithm has completed and the optimization problem is solved. Therefore, the term B can *not* be made zero simply by preprocessing the data and removing its mean. This is the primary reason why having a poor margin is unavoidable when the cross-entropy loss is used for low-dimensional data sets.

The statement of Theorem 2 requires that the support vectors lie exactly on an affine subspace. This condition is relaxed in Corollary 2.

Corollary 2 *Assume that the points $\{x_i\}_{i\in\mathcal{I}}$ and $\{x_j\}_{j\in\mathcal{J}}$ in $\mathbb{R}^n$ are linearly separable and there exist a set of orthonormal vectors $\{r_k\}_{k\in K}$ and a set of scalars $\{\Delta_k\}_{k\in K}$ such that*

$$\langle r_k, x_i\rangle \geq \Delta_k,\ \langle r_k, x_j\rangle \leq \Delta_k \quad \forall i \in \mathcal{I}_{sup},\ \forall j \in \mathcal{J}_{sup},\ \forall k \in K.$$

Let $\langle \overline{w}, \cdot\rangle + B = 0$ denote the decision boundary obtained by minimizing the cross-entropy loss, as in Theorem 2. Then the minimization of the cross-entropy loss yields a margin smaller than or equal to

$$\frac{1}{\sqrt{B^2 \sum_{k\in K} \Delta_k^2}}.$$

Proof See Appendix B. ■

Lastly, one could question why the weight parameters are not forced to be bounded in the analysis of this section by adding a penalty term into the training loss function in terms of some norm of the parameters. The short answer is that this section, along with all other parts of this work, aims to reveal how the richness of the training data arises within the dynamics of the gradient descent algorithm, and how it provides a guarantee for robustness. When the cost function is cross-entropy loss, there is no clear relationship between penalizing some norm of the weight parameters and improving the richness of the training data set, and this is why the inclusion of some regularization term, with its classical understanding, is avoided in the analysis of this section. This is clarified in Section 5.1 in detail.

3.2. Input-Based Lipschitz Bounds for Two-Layer Neural Networks

When a two-layer neural network is trained via the gradient descent algorithm, the dynamics of the algorithm will be nonlinear. If, in addition, the cost function used for training is the cross-entropy loss, the parameters of the network will grow unboundedly — provided that there are enough parameters to classify every training point correctly with a proper initialization (Zhang et al., 2017). However, most analysis tools for nonlinear dynamical systems primarily focus on the behaviors of these systems around their equilibria, which are the fixed points of their dynamics, or their limiting behaviors inside bounded regions (Khalil, 1996; Sastry, 2013). Therefore, the common tools in nonlinear analysis will become inapplicable for the analysis of the cross-entropy loss. In this case, instead of studying the convergence of the actual values of the parameters, we can analyze the convergence of the ratio of the parameters. This leads us to consider an alternative concept of convergence: convergence in direction.

Definition ***(Convergence in direction)*** *Given a set of functions, $W_k : [0,\infty) \mapsto \mathbb{R}^{m_k \times n_k}$ for $k \in [L]$, assume $\lim_{t\to\infty} \|W_k(t)\|_F = \infty$ for some $k \in [L]$. The set $(W_1, \ldots, W_L)$ is said to converge in direction to $(\overline{W}_1, \ldots, \overline{W}_L)$ if*

$$\lim_{t\to\infty} \frac{\|W_k(t) - h(t)\overline{W}_k\|_F}{h(t)} = 0 \quad \forall k \in [L]$$

for some function $h : [0,\infty) \mapsto (0,\infty)$ diverging to $+\infty$, where $\overline{W}_k \in \mathbb{R}^{m_k \times n_k}$ is a matrix with bounded elements for each $k \in [L]$, and $\overline{W}_k \neq \mathbf{0}$ at least for some $k \in [L]$.

With this concept of convergence, we are ready to state the next theorem.

Theorem 3 *Assume that a two-layer neural network is trained to classify the sets $\{x_i\}_{i\in\mathcal{I}}$ and $\{x_j\}_{j\in\mathcal{J}}$ by minimizing the cross-entropy loss*

$$\sum\nolimits_{i\in\mathcal{I}} \log\left(1 + e^{-w^\top (Vx_i+b)_+}\right) + \sum\nolimits_{j\in\mathcal{J}} \log\left(1 + e^{w^\top (Vx_j+b)_+}\right)$$

via the continuous-time gradient descent algorithm, where $w \in \mathbb{R}^r$, $V \in \mathbb{R}^{r\times n}$ and $b \in \mathbb{R}^r$. Assume all the training points are correctly classified[2]*, and the parameters (w, V, b) converge in direction to $(\overline{w}, \overline{V}, \overline{b})$ as defined above. Let $(\overline{w}, \overline{V}, \overline{b})$ be scaled such that*

$$\overline{w}^\top(\overline{V}x_i + \overline{b})_+ \geq 1 \quad \forall i \in \mathcal{I}, \tag{6a}$$

$$\overline{w}^\top(\overline{V}x_j + \overline{b})_+ \leq -1 \quad \forall j \in \mathcal{J} \tag{6b}$$

with either equality holding for at least one point. Let $\mathcal{I}_{sup}$ and $\mathcal{J}_{sup}$ denote the points that achieve equality in (6a) and (6b), respectively. Then the Lipschitz constant of the mapping $x \mapsto \overline{w}^\top(\overline{V}x + \overline{b})_+$ is upper bounded by

$$\frac{n_{\text{node}}^{\text{min}}\sqrt{n_{\text{sup}}^{\text{max}}}}{\sqrt{\underline{\lambda}}},$$

where

- *$n_{\text{sup}}^{\text{max}}$ is the maximum number of support vectors that activate the same hidden layer node:*

$$n_{\text{sup}}^{\text{max}} = \max_{k\in[r]} \max_{S\subseteq \mathcal{I}_{sup}\cup\mathcal{J}_{sup}} \left\{|S| : \overline{V}_k x_s + \overline{b}_k > 0 \, \forall s \in S\right\},$$

- *$n_{\text{node}}^{\text{min}}$ is the minimum number of nodes that are activated by all of the support vectors:*

$$n_{\text{node}}^{\text{min}} = \min_{K\subseteq[r]} \left\{|K| : \max\nolimits_{k\in K} \overline{V}_k x_s + \overline{b}_k > 0 \, \forall s \in \mathcal{I}_{sup} \cup \mathcal{J}_{sup}\right\},$$

- *$\underline{\lambda}$ is a lower bound for the minimum eigenvalue of $X_k^\top X_k$, where the columns of X_k are the support vectors activating node k.*

Proof See Appendix C. ■

Theorem 3 reveals a new notion of richness for the data set that is markedly different than the full-rankness of the data or of the support vectors; it is the linear independence of the support vectors. If the support vectors activating any one of the hidden-layer nodes are linearly dependent, then $\underline{\lambda}$ in Theorem 3 will be zero, and the bound will become void. Note that this notion of richness also reinforces the result of Theorem 2.

[2] State-of-the-art neural networks are able to achieve zero training error even on randomly generated and randomly labeled data sets (Zhang et al., 2017).

4. Low-Dimensional Features on Multiple Layers

As seen in Section 2 and Section 3, robustness guarantees for neural networks seem to require certain amount of richness in the training data set. While the full rankness of the training data activating each hidden-layer node provides persistency of excitation for a network trained with the squared-error loss, the use of cross-entropy loss necessitates the linear independence of all support vectors activating the same hidden-layer nodes. These two conditions are different, but both of them are likely to be violated if the training data is large in number and the features of the data are low-dimensional in some layer of the network. Even though this appears to be a degenerate case, there are two main reasons why it is also the prevalent case:

1. The raw image, audio, and video data are low-dimensional by nature; this is the fundamental fact that has enabled data compression for decades by discovering and utilizing low-dimensional representations specific to different applications.

2. Even if the input data is full rank, the use of the gradient descent algorithm for training multiple-layer networks induces low-rank signals in the intermediate layers of the network.

The latter item has been observed in several empirical works, for example, in (Martin and Mahoney, 2019). The following theorem confirms its validity for deep linear networks.

Theorem 4 *Assume that a deep linear network with L layers is trained to classify the points in the sets $\{x_i\}_{i\in\mathcal{I}}$ and $\{x_j\}_{j\in\mathcal{J}}$ by minimizing*

$$\ell(W_1,\dots,W_L)=\sum\nolimits_{s\in\mathcal{I}\cup\mathcal{J}} d(W_L\cdots W_1x_s,y_s)+\sum\nolimits_{k\in[L]}\mu_k\|W_k\|_F^2 \tag{7}$$

where $\{y_s : s\in\mathcal{I}\cup\mathcal{J}\}$ is the set of labels, $d(\cdot,\cdot)$ is any loss function differentiable in its first argument, $W_k\in\mathbb{R}^{n_k\times n_{k-1}}$ for all $k\in[L]$ and $n_L=1$, i.e., the output of the network is scalar, and $\mu_k>0$ for all $k\in[L]$. If the gradient descent algorithm converges from a random initialization to a solution $(\overline{W}_1,\dots,\overline{W}_L)$, then each weight matrix $\overline{W}_k$ has rank 1, almost surely.

Proof See Appendix D. ■

Theorem 4 shows that adding the frobenious norms of the weight parameters, which is commonly referred to as adding weight-decay, causes all signals in the hidden layers to have rank 1 — independently of the loss function used for training. The following theorem shows that when the cross-entropy loss is used, the result is the same even without the addition of weight-decay.

Theorem 5 *Assume that the sets $\{x_i\}_{i\in\mathcal{I}}$ and $\{x_j\}_{j\in\mathcal{J}}$ are separable by a hyperplane passing through the origin, and a deep linear network with L layers is trained to classify them by minimizing the cross-entropy loss:*

$$\ell(W_1,\dots,W_L)=\sum\nolimits_{i\in\mathcal{I}}\log\left(1+e^{W_L\cdots W_1x_i}\right)+\sum\nolimits_{j\in\mathcal{J}}\log\left(1+e^{-W_L\cdots W_1x_j}\right) \tag{8}$$

via the continuous-time gradient descent algorithm:

$$\frac{dW_k(t)}{dt}=-\frac{\partial\ell(W_1(t),\dots,W_L(t))}{\partial W_k(t)}\qquad\forall k\in[L].$$

If the weight matrices converge in direction to $(\overline{W}_1,\dots,\overline{W}_L)$ from a random initialization, then each $\overline{W}_k$ has rank 1 almost surely.

Proof See Appendix E. ■

Theorem 4 and Theorem 5 show that the conditions given in the previous sections for the persistency of excitation are easily violated when the model has multiple layers and the gradient descent algorithm is used for training. This suggests that adding more training points into the data set will not be an effective approach to provide persistency of excitation for the parameters of a neural network. The following section tries to address this issue by inserting exogenous disturbance to the dynamics of the gradient descent algorithm in order to ensure persistency of excitation for multiple-layer networks.

5. Persistent Excitation of Parameters for Robustness

It seems that multi-layer architectures make it challenging to ensure persistency of excitation, and this causes lack of robustness in multi-layer neural networks. One straightforward way to remedy this problem is to inject exogenous perturbations into the dynamics of the gradient descent algorithm so that even the parameters in the hidden layers receive persistent excitation during training.

In order to understand how this external excitation should be applied, this section will first look into the traditional procedure to introduce robustness for linear models, which is the addition of regularization term into the loss function, and describe how it could be reframed as a method to ensure persistent excitation of the model parameters. Then the second part of the section introduces an analogous procedure to persistently excite every parameter in a multiple-layer network.

5.1. Reinterpreting Regularization as a Means to Ensure Persistency of Excitation

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 linear regression problem with the squared-error loss:

$$\min_w \sum\nolimits_{i\in\mathcal{I}} (w^\top x_i - y_i)^2. \tag{9}$$

For this problem to have a unique solution, the set of training points $\{x_i\}_{i\in\mathcal{I}}$ needs to span the whole input space, $\mathbb{R}^n$. If the training data set is rank deficient, the usual remedy is to add a regularization term to the cost function (Hastie et al., 2009):

$$\min_w \sum\nolimits_{i\in\mathcal{I}} (w^\top x_i - y_i)^2 + \mathcal{R}(w).$$

The regularization term $\mathcal{R}(w)$ is usually assumed to impose some prior knowledge about w, and it is chosen as a convex function of some norm of w, such as $\|w\|_2^2$ or $\|w\|_1$. This forces the parameters to be close to zero in some norm and restricts the class of functions that can be estimated by the model.

For the ridge regression problem corresponding to the choice $\mathcal{R}(w) = \lambda\|w\|_2^2$ for any $\lambda > 0$:

$$\min_w \sum\nolimits_{i\in\mathcal{I}} (w^\top x_i - y_i)^2 + \lambda\|w\|_2^2, \tag{10}$$

it is an ordinary observation that adding the regularization term is equivalent to adding n ghost measurements, since we have

$$\lambda\|w\|_2^2 = \sum\nolimits_{i=1}^n \left(w^\top(\sqrt{\lambda}e_i) - 0\right)^2,$$

where $e_i \in \mathbb{R}^n$ is the i-th standard basis vector of $\mathbb{R}^n$. Alternatively, an equivalent problem could also be formulated as

$$\min_w \mathbb{E}_{\{\xi_i\}_{i\in\mathcal{I}}} \sum\nolimits_{i\in\mathcal{I}} \left(w^\top(x_i+\xi_i)-y_i\right)^2 \tag{11}$$

where $\xi_i \in \mathbb{R}^n$ is any random vector with

$$\mathbb{E}[\xi_i]=0, \quad \mathbb{E}[\xi_i\xi_i^\top]=\frac{\lambda}{|\mathcal{I}|}I$$

for each $i \in \mathcal{I}$. In particular, the probability distributions for $\{\xi_i\}_{i\in\mathcal{I}}$ could be chosen to be discrete for practical purposes, and the ridge regression could be considered as augmenting the training data set by perturbing each training point in, for example, $2n$ different directions and creating $2n$ new training points, and consequently, producing a new training data set with $2n|\mathcal{I}|$ points in total.

The equivalence of problems (10) and (11) is critical: even though the common regularization terms in use are originally introduced to impose a prior knowledge about the weight parameters and to bound the class of estimated functions, they could have also been introduced as a way to create a larger data set from the original training points and solve the optimization problem with this larger data set. This equivalence seems to depend on the choice of squared ℓ_2 norm as the regularization term, but the following theorem suggests another way to augment the data set for linear regression, which preserves the equivalence for a much larger class of regularization terms.

Theorem 6 *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:*

$$\min_w \sum_{i\in\mathcal{I}}(y_i - w^\top x_i)^2 + \lambda\|w\|_p^m, \tag{12a}$$

$$\min_w \sum_{i\in\mathcal{I}} \frac{1}{2}\left(y_i - \min_{d:\|d\|_q\le\epsilon} w^\top(x_i+d)\right)^2 + \frac{1}{2}\left(y_i - \max_{d:\|d\|_q\le\epsilon} w^\top(x_i+d)\right)^2, \tag{12b}$$

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[3]*. For each λ, there exists some ϵ such that the solutions of the two problems are identical. Conversely, for each ϵ, there exists some λ such that the solutions of the two problems are identical.*

Proof See Appendix F. ■

Note that problem (12b) solves the regression problem with the two extreme points on the q-norm ball around each training point, where the radius of the ball is given by ϵ. This provides a way to ensure that every point inside the ball around x_i is assigned a value close to y_i at the end of optimization — thereby indirectly solving the regression problem with a richer data set while avoiding the generation of a large set of points as in (11).

Theorem 6 shows that the following are equivalent **for linear regression with squared-error loss**:

1. bounding or penalizing any norm of the weight parameters,

[3] The case for higher-dimensional target values is stated in Corollary 3.

2. augmenting the training data set by inflating the training points and solving the regression problem with this larger data set,
3. inducing generalization, i.e., providing comparable results on the training data and on unseen points that are close to the training data.

It is clear that items 2 and 3 are equivalent irrespective of the loss function used for regression. Equivalence of item 1 to the others, however, does depend on the loss function. To see this, consider the cross-entropy loss with two sets of points $\{x_i\}_{i\in\mathcal{I}}$ and $\{x_j\}_{j\in\mathcal{J}}$:

$$\min_w \sum_{i\in\mathcal{I}} \log\left(1+e^{-w^\top x_i}\right) + \sum_{j\in\mathcal{J}} \log\left(1+e^{w^\top x_j}\right),$$

and assume that these sets are linearly separable. If we inflate the training data as in problem (12b), we obtain

$$\begin{aligned}\min_w \; \Bigg\{ &\frac{1}{2}\sum_{i\in\mathcal{I}} \log\left(1+e^{-w^\top x_i-\epsilon\|w\|_q}\right) + \log\left(1+e^{-w^\top x_i+\epsilon\|w\|_q}\right) \\ &+\frac{1}{2}\sum_{j\in\mathcal{J}} \log\left(1+e^{w^\top x_j+\epsilon\|w\|_q}\right) + \log\left(1+e^{w^\top x_j-\epsilon\|w\|_q}\right)\Bigg\}\end{aligned} \tag{13}$$

where ϵ is the p-norm of the disturbance applied to each training point. This problem is not convex, but more importantly, this problem does not attain its global minimum at a finite point — as long as ϵ does not exceed the maximum p-norm margin of the original data set. This is because the two sets remain linearly separable even after each point is inflated by ϵ amount in p-norm. Consequently, the normal vector of any hyperplane separating these inflated points gives a direction for w to drive the cost function arbitrarily close to zero by increasing $\|w\|_q$. Note that the infimum of the cost function is zero, and any bounded vector will be strictly suboptimal.

Problem (13) gives an example of the case where the optimal solution is unbounded but generalizes perfectly for the unseen data near the training points. Moreover, it shows that enforcing boundedness of the parameters could prevent the discovery of the optimal solution. This simple example reveals why **the generalization of the performance of a model does not necessarily require, or entail, the boundedness of the model parameters**. This fact is the main reason why the analyses in Section 3 do not involve any penalty term to impose a bound on the model parameters.

Lastly, it should be noted that problem (12b) is different from adversarial training (Madry et al., 2018). Adversarial training regards the loss function used in the empirical risk minimization as the ultimate goal and tries to minimize this loss function under worst disturbances:

$$\min_w \sum_{i\in\mathcal{I}} \max_{d:\|d\|_p\le\epsilon} \ell(w, x_i+d).$$

This overlooks the fact that the loss function is only a proxy for the actual task. Problem (12b), on the other hand, tries to enforce that the estimate values be close to the target values for all neighbors of the training data — which is the very property needed for robustness. In the next subsection, the formulation of problem (12b) will form our basis for generating persistently exciting perturbations for neural networks.

5.2. A Training Scheme for Persistent Excitation of Neural Network Parameters

Problem (12b) 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 provides persistency of excitation for the parameters of the model. An analogous way to excite the parameters of a multiple-layer neural network could be inserting perturbation to every signal preceding an affine operation inside the network. To illustrate, consider an L-layer feedforward network:

$$h_0(x) = x, \tag{14a}$$

$$h_j(x) = \sigma_j(W_j h_{j-1}(x) + b_j) \quad j = 1, 2, \ldots, L, \tag{14b}$$

where $\{W_j\}_{j\in[L]}$ and $\{b_j\}_{j\in[L]}$ are the weight and bias parameters of the network, and $\{\sigma_j(\cdot)\}_{j\in[L]}$ are the nonlinear activation functions. 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

$$\tilde{h}_0(x; d) = x, \tag{15a}$$

$$\tilde{h}_j(x; d) = \sigma_j(W_j[\tilde{h}_{j-1}(x; d) + d_j] + b_j) \quad j = 1, 2, \ldots, L, \tag{15b}$$

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

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

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

It might seem curious that we perturb the hidden layers as well only to guarantee robustness against perturbations in the input of the network. To reiterate the necessity for this, reconsider the neural network in (14) as composition of two mappings:

$$h_L(x) = (h_{L\leftarrow m} \circ h_m)(x)$$

where L is the total number of layers with $L \geq 3$, m is the depth of an intermediate layer satisfying $2 \leq m \leq L-1$, and $h_{L\leftarrow m}$ represents the mapping from the output of the m-th layer of the network to the output of the last layer. As we have discussed in Section 4, assume that the training data set $\{x_i\}_{i\in\mathcal{I}}$ lies in a low dimensional subspace within the input space. Without loss of generality, let

$$\langle x_i, e_1\rangle = \langle x_i, e_2\rangle = 0 \quad \forall i \in \mathcal{I},$$

where e_1 and e_2 are the first and the second standard basis vectors of $\mathbb{R}^n$. Assume hypothetically that the training procedure causes one of the nodes in the m-th layer to output the product of the first and the second coordinates of the input points. Then for all the training points, this node will output zero, and how the output of this node is propagated in $h_{L\leftarrow m}$ will not be observed at all. Now assume that in an attempt to provide robustness, we only perturb the input points but not the signals in the intermediate layers. If the allowed perturbations are bounded by ϵ in ℓ_∞ norm, the output of the node of interest will be in the range $[-\epsilon^2, \epsilon^2]$, and this will likely not suffice to observe the effect of this node on the output of the network. Consequently, even if the output of the network is predicted well for all the training data and the points in their ϵ-neighborhood, the reaction of the network to the points not in the close neighborhood of the training data will be little known. This argument is well supported by the observations in recent works, which show that training a neural network by (adversarially) perturbing the training data provides a large margin for the training data, but not for the unseen test data (Madry et al., 2018).

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; a batch of training points can also be used at every step to compute the gradient update.

Algorithm 1 Training with Persistent Excitation

```
input: training data $\{x_i\}_{i\in\mathcal{I}}$,
       target values $\{y_i\}_{i\in\mathcal{I}}$,
       neural network $f_\theta(x;d) \equiv \tilde{h}_L(x;d)$ in (15),
       set of allowed perturbations $\mathcal{D}$,
       learning rate $\eta$,
       momentum $\gamma$
output: network parameters $\theta$
initialize: $\Delta\theta \leftarrow 0$
repeat
    randomly choose $i \in \mathcal{I}$
    $d_1 \leftarrow \mathrm{argmax}_{d\in\mathcal{D}}\, f_\theta(x_i;d)$
    $d_2 \leftarrow \mathrm{argmin}_{d\in\mathcal{D}}\, f_\theta(x_i;d)$
    $\Delta\theta \leftarrow \gamma\Delta\theta + (1-\gamma)\nabla_\theta\left[(f_\theta(x_i;d_1)-y_i)^2 + (f_\theta(x_i;d_2)-y_i)^2\right]$
    $\theta \leftarrow \theta - \eta\Delta\theta$
until training is complete
```

6. Experimental Results

In this section, we test Algorithm 1 on a binary classification task. Only two classes of images, the horses and the planes, have been chosen from the CIFAR-10 data set for the classification task. The same network architecture is used in all of the experiments: two convolutional layers followed by two fully-connected layers with leaky-ReLu activations. Neither batch-normalization nor drop-out is implemented in the experiments. The reason why we test Algorithm 1 on a binary classification task and how it could be extented to multiple-class classification tasks are discussed in Section 7.

The first experiment is to demonstrate that classical training paradigm with the squared-error loss and the cross-entropy loss is not effective in providing a large margin for the unseen data points. A four-layer network is trained with these loss functions with no exogenous perturbation for persistency of excitation. The plots in Figure 4 show the percentage of points the network misclassify versus the amount of disturbance needed in the input of the network to cause the misclassification, which is computed by using the projected gradient attack algorithm (Madry et al., 2018; Rauber et al., 2017). In this sense, the plots are analogous to the cumulative distribution function of the margin of the data; and the lower the plot, the more robust the network. It is observed that neither the squared-error loss nor the cross-entropy loss is effective in providing a large margin for the unseen test data.

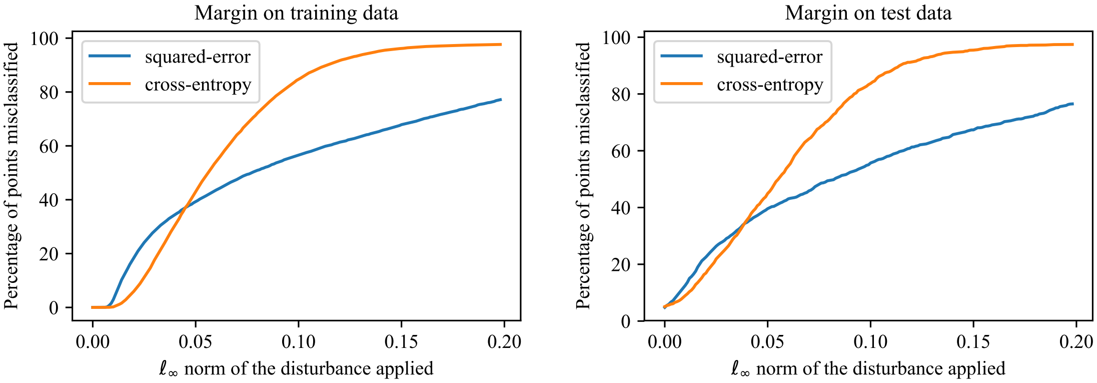

Figure 4: A four-layer network is trained for a binary-classification task with the cross-entropy and the squared-error loss. The plots demonstrate the percentage of points the network misclassifies versus the amount of disturbance needed in the input of the network to cause the misclassification. The curves are analogous to the cumulative distribution function of the margin between the decision boundary of the classifier and the data set. The lower the curve, the more robust the network. The plots show that neither the squared-error loss nor the cross-entropy loss is effective in providing a large margin for the unseen test data. Note that a perturbation of magnitude 1.0 changes a white pixel to an exact gray.

The next experiment is to see the effectiveness of Algorithm 1. We train the same four-layer network with Algorithm 1 by using three different sets of perturbation, which are ℓ_∞ balls with radii 0.005, 0.010 and 0.020 on all layers. Comparing the left plot of Figure 4 with the left plot of Figure 5, we observe that applying a small persistently-exciting perturbation increases the margin of the training data conspicuously. We also observe that the training error is not able to reach zero when the perturbation is relative large, as shown by the curve for 0.020 perturbation. Nevertheless, the same magnitude of perturbation attains the largest margin on the test data.

The last experiment is to demonstrate the necessity of perturbing all layers, and not only the first layer, in order to ensure persistent excitation of the parameters of the network. Figure 6 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 are restricted to be in ℓ_∞ ball with radius 0.020. 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 for persistency of excitation as described in Algorithm 1, the margin of the training data becomes a good indicator of the margin of the test data.

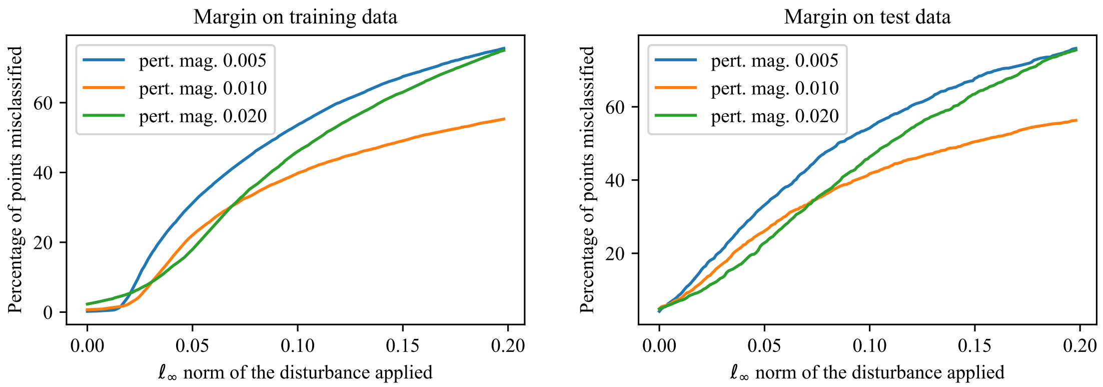

Figure 5: The effect of training with persistent excitation as described in Algorithm 1. Small perturbations for persistent excitation improves the margin for the training data. The larger pertubations prevent the training error from reaching zero; however, they yield larger margins on the test data.

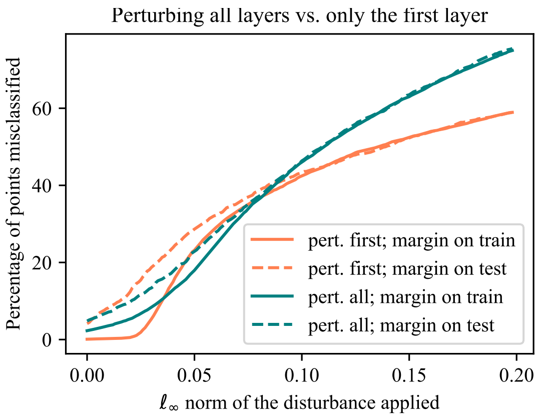

Figure 6: The same network is trained in two different ways: by perturbing every layer of the network as described in Algorithm 1 for persistency of excitation, and by perturbing only the training data (the first layer) similar to adversarial training. Perturbing only the first layer substantially increases the margin for the training data; however, this is not reflected in the test data. In contrast, when all layers are perturbed for persistency of excitation as outlined in Algorithm 1, the margin of the training data becomes a good indicator of the margin of the test data.

7. Discussion

In this section, we discuss why we have restricted our analyses mostly to binary classification, how it could be extended to multiple-class classification, how Algorithm 1 could be applied to different loss functions, and how its performance could be improved. We also compare our key results with the previous works.

7.1. Emphasis on Binary Classification and Extensions to Multi-Class Classification

In this work, when studying the robustness of neural networks, we have focused on binary classification tasks. There are two main reasons for this.

1. Consider the subject of hypothesis testing. The analysis of binary hypothesis testing naturally precedes the study of multiple hypotheses, and it is essential to reveal the critical factors and the bottlenecks in hypothesis testing, such as the trade-off between Type-I and Type-II errors, the ROC curve, and the Neyman-Pearson rule (Poor, 2013; Keener, 2011). Similarly, the study of robustness of machine learning models must start with the simplest case where robustness can be analyzed clearly if we truly want to understand what causes lack of robustness and how it can be addressed. This simplest case appears to be the binary classification.

2. Binary classification is used in many critical tasks. For example, given the medical data of a patient, the decision of whether the patient has a certain disease or not is a binary classification task. Similarly, in computer security, the decision of whether a certain face or fingerprint belongs to the set of authorized users is a binary classification task.

There are two straighforward ways to extend Algorithm 1 to the classification of multiple classes. The first approach is to use a hierarchy of binary classifiers. Given m different classes, each training point can be assigned to a class with at most $\lceil \log_2(m) \rceil$ binary classifiers. The performance of this classifier depends on the choice of ordering and the groups of classes.

The second approach avoids using a hierarchy of classifiers and tries to impose fairness by employing equidistant target values inside a higher dimensional space. This approach is based on the following corollary to Theorem 6.

Corollary 3 *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 $\mathbb{R}^m$, consider the following two problems:*

$$\min_{W} \quad \sum_{i\in\mathcal{I}} \|y_i - Wx_i\|_2^2 + \lambda \|W\|_{p_1,p_2}^2 \tag{16}$$

and

$$\min_{W} \quad \sum_{i\in\mathcal{I}} \|y_i - W(x_i + d_i)\|_2^2 + \|y_i - W(x_i - d_i)\|_2^2 \tag{17a}$$

$$\text{subject to} \quad d_i \in \operatorname*{argmax}_{d:\|d\|_{p_1}\le\epsilon} \max_{v:\|v\|_{q_2}\le 1} \left\{ v^\top W(x_i + d) - v^\top W x_i \right\} \quad \forall i \in \mathcal{I}, \tag{17b}$$

where $\|\cdot\|_{p_2}$ and $\|\cdot\|_{q_2}$ are dual norms, and $\|W\|_{p_1,p_2}$ denotes the induced norm of W:

$$\|W\|_{p_1,p_2} = \max_{u:\|u\|_{p_1}=1} \|Wu\|_{p_2}.$$

For each λ, there exists some ϵ such that the solutions of the two problems are identical. Conversely, for each ϵ, there exists some λ such that the solutions of the two problems are identical. ■

The involvement of x_i in (17b) might seem unnecessary since the model is linear, but remember that we introduce this corollary to find persistently exciting perturbations for nonlinear classifiers. To demonstrate the use of Corollary 3 for a multi-class classifier in an example, assume that the training data $\{x_i\}_{i\in\mathcal{I}}$ in $\mathbb{R}^n$ come from four different classes, and the labels for these classes are $\{y_i\}_{i\in\mathcal{I}}$. For these labels, we can associate four equidistant vectors $z^{(1)}, z^{(2)}, z^{(3)}, z^{(4)} \in \mathbb{R}^3$ such that

$$\begin{aligned} y_i &\in \{z^{(1)}, z^{(2)}, z^{(3)}, z^{(4)}\} && \forall i \in \mathcal{I}, \\ \|z^{(i)} - z^{(j)}\|_2 &= 1 && \forall i, j \in \{1,2,3,4\},\ i \neq j. \end{aligned}$$

Then the problem of training a classifier $f_\theta : \mathbb{R}^n \to \mathbb{R}^3$ with squared-error loss can be formulated as

$$\min_\theta \sum\nolimits_{i\in\mathcal{I}} \|f_\theta(x_i) - y_i\|_2^2 .$$

Corresponding to point x_i, we can find two different persistently-exciting perturbations, d_{i_1} and d_{i_2}, by solving

$$\begin{aligned} v_i &= \underset{v:\|v\|_2\leq 1}{\text{argmax}} \max_{d\in\mathcal{D}} \left\{ v^\top f_\theta(x_i; d) - v^\top f_\theta(x_i) \right\}, \\ d_{i_1} &\in \underset{d\in\mathcal{D}}{\text{argmax}}\ v_i^\top f_\theta(x_i; d), \\ d_{i_2} &\in \underset{d\in\mathcal{D}}{\text{argmin}}\ v_i^\top f_\theta(x_i; d), \end{aligned}$$

where $f_\theta(x; d)$ is the network represented by $\tilde{h}_L(x; d)$ in (15). With these perturbations, we can write the optimization problem with persistent excitation as

$$\min_\theta \sum\nolimits_{i\in\mathcal{I}} \|f_\theta(x_i; d_{i_1}) - y_i\|_2^2 + \|f_\theta(x_i; d_{i_2}) - y_i\|_2^2.$$

Given the solution $\hat{\theta}$ to this problem, we can finally describe the classifier as

$$x \mapsto \text{argmin}_{z\in\{z^{(1)},z^{(2)},z^{(3)},z^{(4)}\}} \|f_{\hat{\theta}}(x) - z\|_2.$$

7.2. Alternative Loss Functions and Perturbation Sets

The proposed algorithm for persistent excitation considers only the squared-error loss function. An analogous approach for the cross-entropy loss is possible, but the perturbations for the hidden layers need to be either not additive or not bounded. This is because the cross-entropy loss yields unbounded growth of the parameters of the network, and consequently, the signals in the hidden layers will become unboundedly large. In this case, the addition of finite perturbations will have no effect on the output of the network. One potential alternative is introducing multiplicative perturbations to the signals in the hidden layers.

In Algorithm 1, the set of allowed perturbations $\mathcal{D}$ is left as an arbitrary set. In our experiments, this set is chosen to be an ℓ_∞ ball with the same radius for all layers. Applying perturbations with different magnitudes to different layers of the network could improve the robustness of the network, while preserving its overall accuracy.

7.3. Related Works

There are three main subjects that are closely related to our results.

1. Logistic regression and maximum margin: Recently there has been a line of work (Soudry et al., 2018; Ji and Telgarsky, 2018) claiming that linear classifiers trained with the gradient descent algorithm achieve the maximum margin if the cross-entropy loss function is used — which sounds contrary to the results in Section 3. There are two reasons for this discrepancy. The first reason is an unfortunate misnomer: support vector machines with no bias term are also called the maximum-margin solution in some sources, e.g. (Rosset et al., 2004; Shalev-Shwartz and Ben-David, 2014), even though the decision boundary found with these problems is forced to pass through the origin, and hence, necessarily suboptimal. The second and the more critical reason is that the analysis in (Soudry et al., 2018) starts with removing the necessity for a bias term by inserting the training data into an augmented space; consequently, all claims about maximum margin are stated for this augmented state. In other words, **this so-called maximum margin is attained in an augmented space; not in the original space the training points lie in**. However, as demonstrated in Figure 2, the true margin between the training data and the decision boundary of the classifier could be substantially smaller than the optimal margin that could be obtained in the original space. This distinction has not been emphasized, nor mentioned, in (Soudry et al., 2018), and the ambiguity has led to the misperception that the gradient descent algorithm on the cross-entropy loss yields the *true* maximum-margin solution in the original space — despite the fact that neural networks are consistently shown to misclassify inputs that are barely different than their *training* data (Goodfellow et al., 2015), and therefore, there is no possibility they could have a large margin between their decision boundary and the training data. Unfortunately, this misconception has been continued and even been used in an attempt to explain the generalization of neural networks (Wei et al., 2019).

2. Importance of low-dimensionality: As discussed in Section 4, most of the data used for training neural networks, such as image, audio or video, lie in a low-dimensional space in some domain. Furthermore, the iterative training algorithms induce low-dimensional signals in the intermediate layers of neural networks independently of the training data. This fact has been critical to draw the conclusion that it is not possible to achieve robustness in neural networks simply by retrieving more training data. If this low-dimensionality is overlooked, the conclusion will be different. Indeed, Schmidt et al. (2018) analyzes the robustness of classifiers with full-rank, non-degenerate data sets and single-layer, linear classifiers, and arrives at the conclusion that the robustness could be achieved with more training data.

3. Trade-off between training speed and generalization: It has been shown in recent works that adaptive gradient algorithms do not usually yield solutions which generalize well, even though they accelerate the training procedure substantially (Wilson et al., 2017). The concept of persistent excitation gives an insight about this: the adaptive gradient methods remove the effect of correctly classified points too quickly and do not allow these points to persistently excite the parameters of the model. The same argument applies to the training loss functions: if a loss function removes the effect of correctly classified points too quickly, then it could yield a worse generalization, as seen in the case of cross-entropy loss.

8. Conclusion

In this work, we looked into the dynamics of the gradient descent algorithm to understand why neural networks lack robustness and how this vulnerability could be addressed. We identified sufficient conditions on the training data sets to ensure robustness of two-layer neural networks. We showed that for classification tasks, multi-layer networks are unlikely to satisfy these conditions, and the signals interacting with the parameter estimates consequently fail to remain persistently exciting. This lack of persistent excitation removes all the guarantees about how the network will react to inputs that are different than its training data.

To provide a solution for this problem, we studied the classical regularization terms used for linear models closely and reframed them in the context of persistency of excitation. Based on this framework, we introduced a new algorithm to generate persistently exciting perturbations and tested this algorithm on a binary classification task. We demonstrated that the proposed algorithm improves the correspondence between the margins of the training data and the test data.

References

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

D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific Optimization and Computation Series. Athena Scientific, 1999.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

Frank M. Callier and Charles A. Desoer. *Linear System Theory*. Springer-Verlag, 2012.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2 edition, 2009.

Ziwei Ji and Matus Telgarsky. Risk and parameter convergence of logistic regression. *arXiv preprint arXiv:1803.07300*, 2018.

Robert W Keener. *Theoretical Statistics: Topics for a Core Course*. Springer, 2011.

H.K. Khalil. *Nonlinear Systems*. Prentice Hall, 1996.

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

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Charles H. Martin and Michael W. Mahoney. Traditional and heavy tailed self regularization in neural network models. In *International Conference on Machine Learning*, 2019.

Kamil Nar and Shankar Sastry. Step size matters in deep learning. In *Advances in Neural Information Processing Systems*, pages 3440–3448, 2018.

H Vincent Poor. *An Introduction to Signal Detection and Estimation*. Springer-Verlag, 2013.

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

Saharon Rosset, Ji Zhu, and Trevor J Hastie. Margin maximizing loss functions. In *Advances in Neural Information Processing Systems*, pages 1237–1244, 2004.

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

Shankar Sastry and Marc Bodson. *Adaptive Control: Stability, Convergence and Robustness*. Prentice Hall, 1989.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pages 5014–5026, 2018.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Eliza Strickland. IBM Watson, heal thyself: How IBM overpromised and underdelivered on AI health care. *IEEE Spectrum*, 56(4):24–31, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

Colin Wei, Jason Lee, Qiang Liu, and Tengyu Ma. On the margin theory of feedforward neural networks, 2019.

Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4148–4158, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Appendix A. Proof of Theorem 1

To begin with, assume b is fixed and not updated by the gradient descent algorithm. Let $(\hat{W}, \hat{V})$ denote the local optimum that the algorithm has converged. For point x_i, let $G_i \in \{0,1\}^{r\times r}$ be the diagonal matrix that satisfies

$$\hat{W}(\hat{V}x_i + b)_+ = \hat{W}G_i(\hat{V}x_i + b).$$

The update rule of the gradient descent algorithm is given as

$$W \leftarrow W - \delta\left\{\sum\nolimits_{i\in\mathcal{I}} \left[WG_i(Vx_i+b) - f(x_i)\right](Vx_i+b)^\top G_i^\top\right\}, \tag{18a}$$

$$V \leftarrow V - \delta\left\{\sum\nolimits_{i\in\mathcal{I}} G_i^\top W^\top \left[WG_i(Vx_i+b) - f(x_i)\right] x_i^\top\right\}, \tag{18b}$$

followed by

$$(G_i)_{kk} \leftarrow \mathbf{1}_{\{V_k x_i + b_k > 0\}} \quad \forall k \in [r],\ \forall i \in \mathcal{I},$$

where $(G_i)_{kk}$ denotes the k-th diagonal element of the matrix G_i, and V_k and b_k represent the k-th rows of V and b. If the algorithm has converged, switching between 1 and 0 must have ended in a finite time. Therefore, around an equilibrium point, we will assume G_i is fixed for all $i \in \mathcal{I}$.

Like all dynamical systems, the gradient descent algorithm (18) can converge to an equilibrium from randomly chosen close neighbors of this point only if the equilibrium is *stable in the sense of Lyapunov* (Sastry, 2013). A necessary condition for an equilibrium of a nonlinear dynamical system to be stable is that the linear approximation of this dynamical system not be unstable around the same equilibrium. We will use this fact to obtain a necessary condition for the algorithm to converge to an equilibrium from any randomly chosen initial point.

Linearization of the system (18) around the equilibrium $(\hat{W}, \hat{V})$ gives

$$(W - \hat{W}) \leftarrow (W - \hat{W}) - f_1(W - \hat{W}) - f_2(V - \hat{V}), \tag{19a}$$

$$(V - \hat{V}) \leftarrow (V - \hat{V}) - f_3(W - \hat{W}) - f_4(V - \hat{V}), \tag{19b}$$

where

$$\begin{aligned}
f_1(\Delta W) &= \delta \sum_{i\in\mathcal{I}} \Delta W G_i(\hat{V}x_i + b)(\hat{V}x_i + b)^\top G_i^\top, \\
f_2(\Delta V) &= \delta \sum_{i\in\mathcal{I}} \hat{W}G_i \Delta V x_i x_i^\top \hat{V} G_i^\top + \delta \sum_{i\in\mathcal{I}} [\hat{W}G_i(\hat{V}x_i+b) - f(x_i)]x_i^\top \Delta V^\top G_i^\top, \\
f_3(\Delta W) &= \delta \sum_{i\in\mathcal{I}} G_i^\top \hat{W}^\top \Delta W G_i(\hat{V}x_i + b)x_i^\top + \delta \sum_{i\in\mathcal{I}} G_i^\top \Delta W^\top [\hat{W}G_i(\hat{V}x_i+b) - f(x_i)]x_i^\top, \\
f_4(\Delta V) &= \delta \sum_{i\in\mathcal{I}} G_i^\top \hat{W}^\top \hat{W} G_i \Delta V x_i x_i^\top.
\end{aligned}$$

We have the equality

$$\langle \Delta W, f_2(\Delta V)\rangle = \langle f_3(\Delta W), \Delta V\rangle \quad \forall \Delta W \in \mathbb{R}^{m\times r},\ \forall \Delta V \in \mathbb{R}^{r\times n};$$

therefore, the linearized system (19) can be represented by a symmetric matrix[4], and the system (19) is stable in the sense of Lyapunov only if the matrix corresponding to the system

$$(W - \hat{W}) \leftarrow (W - \hat{W}) - f_1(W - \hat{W}) \tag{20a}$$

$$(V - \hat{V}) \leftarrow (V - \hat{V}) - f_4(V - \hat{V}) \tag{20b}$$

has all of its eigenvalues inside the unit circle. This implies that the convergence of the system (19) requires the largest eigenvalues of the mappings $f_1(\cdot)$ and $f_4(\cdot)$ to be smaller than 2.

The largest eigenvalue of the mapping $f_1(\cdot)$ being smaller than 2 implies that

$$e_k^\top \sum\nolimits_{i\in\mathcal{I}} G_i(\hat{V}x_i + b)(\hat{V}x_i + b)^\top G_i^\top e_k \le \frac{2}{\delta} \quad \forall k \in [r],$$

where e_k denotes the standard basis vector with 1 in its k-th element. Then,

$$e_k^\top \sum\nolimits_{i\in\mathcal{I}_k} (\hat{V}x_i + b)(\hat{V}x_i + b)^\top e_k \le \frac{2}{\delta} \quad \forall k \in [r]$$

where $\mathcal{I}_k$ denotes the set of indices of the points that activate node k at equilibrium, i.e.,

$$\mathcal{I}_k = \{i \in \mathcal{I} : e_k^\top(\hat{V}x_i + b) > 0\}.$$

Let $\hat{V}_k$ and b_k denote the k-th rows of $\hat{V}$ and b. Then we need

$$\sum\nolimits_{i\in\mathcal{I}_k} (\hat{V}_k x_i + b_k)(x_i^\top \hat{V}_k^\top + b_k) \le \frac{2}{\delta},$$

or equivalently,

$$\hat{V}_k^\top \left(\sum\nolimits_{i\in\mathcal{I}_k} x_i x_i^\top\right) \hat{V}_k + 2b_k \hat{V}_k^\top \left(\sum\nolimits_{i\in\mathcal{I}_k} x_i\right) + b_k^2 - \frac{2}{\delta} \le 0.$$

This is a quadratic inequality in $\hat{V}_k$, and the largest value $\left\|\hat{V}_k\right\|_2$ can take is upper bounded by the larger root of

$$\lambda_{\text{min}}\left(\sum\nolimits_{i\in\mathcal{I}_k} x_i x_i^\top\right) \left\|\hat{V}_k\right\|_2^2 - 2\,|b_k| \left\|\sum\nolimits_{i\in\mathcal{I}_k} x_i\right\|_2 \left\|\hat{V}_k\right\|_2 + b_k^2 - \frac{2}{\delta} = 0.$$

Therefore, we have

$$\left\|\hat{V}_k\right\|_2 \le \frac{|b_k|\,\mu_k}{\lambda_k^{\text{min}}} + \frac{1}{\lambda_k^{\text{min}}}\sqrt{b_k^2(\mu_k)^2 + \lambda_k^{\text{min}}\left(\frac{2}{\delta} - b_k^2\right)}$$

where

$$\mu_k = \left\|\sum\nolimits_{i\in\mathcal{I}_k} x_i\right\|_2,$$

$$\lambda_k^{\text{min}} = \lambda_{\text{min}}\left(\sum\nolimits_{i\in\mathcal{I}_k} x_i x_i^\top\right).$$

[4] This is also a direct consequence of the fact that f_2 and f_3 are the Jacobians of the gradient of the training loss function with respect to the same parameters in different orders.

As an upper bound independent of k, we can write

$$\left\|\hat{V}_k\right\|_2 \leq \frac{\overline{\mu}\|b\|_\infty}{\underline{\lambda}} + \sqrt{\frac{\overline{\mu}^2\|b\|_\infty^2}{\underline{\lambda}^2} + \frac{1}{\underline{\lambda}}\left(\frac{2}{\delta} - \|b\|_\infty^2\right)} \qquad \forall k \in [r]$$

where $\overline{\mu} = \max_{k\in[r]} \mu_k$ and $\underline{\lambda} = \min_{k\in[r]} \lambda_k^{\text{min}}$.

So far we have only used the fact that the largest eigenvalue of $f_1(\cdot)$ is less than 2. Similarly, the largest eigenvalue of $f_4(\cdot)$ being less than 2 implies that

$$\sum\nolimits_{i\in I_k} e_k^\top \hat{W}^\top \hat{W} e_k \cdot \lambda_{\text{min}}\left(\sum\nolimits_{i\in\mathcal{I}_k} x_i x_i^\top\right) \leq \frac{2}{\delta}.$$

If $\hat{W}_k$ denotes the k-th column of $\hat{W}$, we have

$$\left\|\hat{W}_k\right\|_2 \leq \sqrt{\frac{2}{\delta\underline{\lambda}}} \qquad \forall k \in [r].$$

Given the estimates $\hat{W}$ and $\hat{V}$, for every $x \in \mathbb{R}^n$, the function estimated by the network is $\hat{W}(\hat{V}x + b)_+$, and the Lipschitz constant of this estimate is bounded by

$$\sum\nolimits_{k=1}^r \left\|\mathbf{1}_{\{\hat{V}_k x + b > 0\}} \hat{W}_k \hat{V}_k\right\|_F,$$

which is further bounded by

$$n_{\text{active}}^{\text{max}} \left(\sqrt{\frac{2}{\delta\underline{\lambda}}}\right)\left(\frac{\overline{\mu}\|b\|_\infty}{\underline{\lambda}} + \sqrt{\frac{\overline{\mu}^2\|b\|_\infty^2}{\underline{\lambda}^2} + \frac{1}{\underline{\lambda}}\left(\frac{2}{\delta} - \|b\|_\infty^2\right)}\right)$$

and

$$n_{\text{active}}^{\text{max}} \left(\sqrt{\frac{2}{\delta\underline{\lambda}}}\right)\left(\frac{2\overline{\mu}\|b\|_\infty}{\underline{\lambda}} + \sqrt{\frac{1}{\underline{\lambda}}\left|\frac{2}{\delta} - \|b\|_\infty^2\right|}\right)$$

where $n_{\text{active}}^{\text{max}}$ is the maximum number of nodes that a point in $\mathbb{R}^n$ can activate, i.e.,

$$n_{\text{active}}^{\text{max}} = \max_{x\in\mathbb{R}^n} \sum\nolimits_{k=1}^r \mathbf{1}_{\{\hat{V}_k x + b > 0\}}.$$

To complete the proof, now assume that b is not fixed and it is also updated by the gradient descent algorithm. We can write the linearization of the update rule for (W, V, b) as

$$\begin{bmatrix} W \\ V^\top \end{bmatrix} \leftarrow \begin{bmatrix} W \\ V^\top \end{bmatrix} - g_1\left(\begin{bmatrix} W \\ V^\top \end{bmatrix}\right) - g_2(b), \tag{21a}$$

$$b \leftarrow b - g_3\left(\begin{bmatrix} W \\ V^\top \end{bmatrix}\right) - g_4(b), \tag{21b}$$

where g_1, g_2, g_3 and g_4 are the linear operators obtained by taking Jacobians of the gradients of the training loss function with respect to W, V and b. Similar to f_2 and f_3 in system (19), the operators g_2 and g_3 are the Hermitian of each other, and therefore, the matrix corresponding to the system (21) is still symmetric. As a result, its eigenvalues are less than 1 in magnitude only if its diagonal sub-blocks have eigenvalues less than 1 in magnitude, which leads to the identical condition for the case with fixed b. ■

Appendix B. Proof of Theorem 2

Lemma 4 (Adapted from Theorem 3 of Soudry et al. 2018) *Given two sets of points $\{x_i : i \in \mathcal{I}\}$ and $\{x_j : j \in \mathcal{J}\}$ that are linearly separable in $\mathbb{R}^n$, let $\tilde{x}_i$ and $\tilde{x}_j$ denote $[x_i^\top\ 1]^\top$ and $[x_j^\top\ 1]^\top$, respectively, for all $i \in \mathcal{I}$, $j \in \mathcal{J}$. Then the iterate of the gradient descent algorithm, $\tilde{w}(t)$, on the cross-entropy loss function*

$$\min_{\tilde{w}\in\mathbb{R}^{n+1}} \sum\nolimits_{i\in\mathcal{I}} \log(1+e^{-\tilde{w}^\top \tilde{x}_i}) + \sum\nolimits_{j\in\mathcal{J}} \log(1+e^{\tilde{w}^\top \tilde{x}_j})$$

with a sufficiently small step size will converge in direction:

$$\lim_{t\to\infty} \frac{\tilde{w}(t)}{\|\tilde{w}(t)\|} = \frac{\overline{w}}{\|\overline{w}\|},$$

where $\overline{w}$ is the solution to

$$\begin{aligned} \underset{z\in\mathbb{R}^{n+1}}{\text{minimize}} \quad & \|z\|^2 \\ \text{subject to} \quad & \langle z, \tilde{x}_i\rangle \geq 1 \quad \forall i \in \mathcal{I}, \\ & \langle z, \tilde{x}_j\rangle \leq -1 \quad \forall j \in \mathcal{J}. \end{aligned} \tag{22}$$

Proof of Theorem 2. Assume that $\overline{w} = u + \sum_{k\in K} \alpha_k r_k$, where $u \in \mathbb{R}^n$ and $\langle u, r_k\rangle = 0$ for all $k \in K$. By denoting $z = [w^\top\ b]^\top$, the Lagrangian of the problem (22) can be written as

$$\frac{1}{2}\|w\|^2 + \frac{1}{2}b^2 + \sum\nolimits_{i\in\mathcal{I}} \mu_i(1 - \langle w, x_i\rangle - b) + \sum\nolimits_{j\in\mathcal{J}} \nu_j(-1 + \langle w, x_j\rangle + b),$$

where $\mu_i \geq 0$ for all $i \in \mathcal{I}$ and $\nu_j \geq 0$ for all $j \in \mathcal{J}$. KKT conditions for the optimality of $\overline{w}$ and B requires that

$$\overline{w} = \sum_{i\in\mathcal{I}} \mu_i x_i - \sum_{j\in\mathcal{J}} \nu_j x_j, \quad B = \sum_{i\in\mathcal{I}} \mu_i - \sum_{j\in\mathcal{J}} \nu_j,$$

and consequently, for each $k \in K$,

$$\begin{aligned} \langle \overline{w}, r_k\rangle &= \sum\nolimits_{i\in\mathcal{I}} \mu_i \langle x_i, r_k\rangle - \sum\nolimits_{j\in\mathcal{J}} \nu_j \langle x_j, r_k\rangle \\ &= \sum\nolimits_{i\in\mathcal{I}} \Delta_k \mu_i - \sum\nolimits_{j\in\mathcal{J}} \Delta_k \nu_j = B\Delta_k. \end{aligned}$$

Then, we can write $\overline{w}$ as

$$\overline{w} = u + \sum\nolimits_{k\in K} B\Delta_k r_k.$$

Let $\langle w_{\text{SVM}}, \cdot\rangle + b_{\text{SVM}} = 0$ denote the hyperplane obtained as the solution of maximum hard-margin SVM in the original space — not in the augmented space. Then w_{SVM} solves

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \|w\|^2 \\ \text{subject to} \quad & \langle w, x_i - x_j\rangle \geq 2 \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}. \end{aligned} \tag{23}$$

Since the vector u also satisfies $\langle u, x_i - x_j\rangle = \langle w, x_i - x_j\rangle \geq 2$ for all $i \in \mathcal{I}, j \in \mathcal{J}$, we have $\|u\| \geq \|w_{\text{SVM}}\| = \frac{1}{\gamma_{\text{OPT}}}$. As a result, the margin obtained by minimizing the cross-entropy loss is

$$\frac{1}{\|\overline{w}\|} = \frac{1}{\sqrt{\|u\|^2 + \sum \|B\Delta_k r_k\|^2}} \leq \frac{1}{\sqrt{\frac{1}{\gamma_{\text{OPT}}^2} + B^2 \sum \Delta_k^2}}.$$

■

Proof of Corollary 2. If $B < 0$, we could consider the hyperplane $\langle \overline{w}, \cdot \rangle - B = 0$ for the points $\{-x_i\}_{i \in \mathcal{I}}$ and $\{-x_j\}_{j \in \mathcal{J}}$, which would have the identical margin due to symmetry. Therefore, without loss of generality, assume $B \geq 0$. As in the proof of Theorem 2, KKT conditions for the optimality of $\overline{w}$ and B requires

$$\overline{w} = \sum_{i \in \mathcal{I}} \mu_i x_i - \sum_{j \in \mathcal{J}} \nu_j x_j, \quad B = \sum_{i \in \mathcal{I}} \mu_i - \sum_{j \in \mathcal{J}} \nu_j$$

where $\mu_i \geq 0$ and $\nu_j \geq 0$ for all $i \in \mathcal{I}, j \in \mathcal{J}$. Note that for each $k \in K$,

$$\begin{aligned} \langle \overline{w}, r_k \rangle &= \sum\nolimits_{i \in \mathcal{I}} \mu_i \langle x_i, r_k \rangle - \sum\nolimits_{j \in \mathcal{J}} \nu_j \langle x_j, r_k \rangle \\ &= B\Delta_k + \sum\nolimits_{i \in \mathcal{I}} \mu_i (\langle x_i, r_k \rangle - \Delta_k) \\ &\quad - \sum\nolimits_{j \in \mathcal{J}} \nu_j (\langle -x_j, r_k \rangle - \Delta_k) \ \geq \ B\Delta_k. \end{aligned}$$

Since $\{r_k\}_{k \in K}$ is an orthonormal set of vectors,

$$\|\overline{w}\|^2 \geq \sum\nolimits_{k \in K} \langle \overline{w}, r_k \rangle^2 \geq \sum\nolimits_{k \in K} B^2 \Delta_k^2.$$

The result follows from the fact that $\|\overline{w}\|^{-1}$ is an upper bound on the margin. ■

Appendix C. Proof of Theorem 3

Lemma 5 *The direction parameters $\overline{w}, \overline{V}$ and $\overline{b}$ satisfy*

$$\|\overline{w}\|_2^2 = \|\overline{V}\|_F^2 + \|\overline{b}\|_2^2.$$

Proof The continuous-time gradient descent algorithm gives

$$\frac{dw}{dt} = \sum_{i \in \mathcal{I}} G_i(Vx_i + b) \frac{e^{-w^\top G_i(Vx_i+b)}}{1 + e^{-w^\top G_i(Vx_i+b)}} - \sum_{j \in \mathcal{J}} G_j(Vx_j + b) \frac{e^{w^\top G_j(Vx_j+b)}}{1 + e^{w^\top G_j(Vx_j+b)}} \tag{24a}$$

$$\frac{dV}{dt} = \sum_{i \in \mathcal{I}} G_i w x_i^\top \frac{e^{-w^\top G_i(Vx_i+b)}}{1 + e^{-w^\top G_i(Vx_i+b)}} - \sum_{j \in \mathcal{J}} G_j w x_j^\top \frac{e^{w^\top G_j(Vx_j+b)}}{1 + e^{w^\top G_j(Vx_j+b)}} \tag{24b}$$

$$\frac{db}{dt} = \sum_{i \in \mathcal{I}} G_i w \frac{e^{-w^\top G_i(Vx_i+b)}}{1 + e^{-w^\top G_i(Vx_i+b)}} - \sum_{j \in \mathcal{J}} G_j w \frac{e^{w^\top G_j(Vx_j+b)}}{1 + e^{w^\top G_j(Vx_j+b)}} \tag{24c}$$

Note that, for all $t \geq 0$ we have

$$\frac{d}{dt}\langle w(t), w(t) \rangle = \frac{d}{dt}\langle V(t), V(t) \rangle + \frac{d}{dt}\langle b(t), b(t) \rangle.$$

Consequently, for all $t \geq 0$:

$$\frac{\|w(t)\|_2^2 - \|w(0)\|_2^2}{h^2(t)} = \frac{\|V(t)\|_F^2 - \|V(0)\|_F^2}{h^2(t)} + \frac{\|b(t)\|_2^2 - \|b(0)\|_2^2}{h^2(t)},$$

which proves that

$$\|\overline{w}\|_2^2 = \|\overline{V}\|_F^2 + \|\overline{b}\|_2^2.$$

■

Lemma 6 *The solution obtained by the continuous-time gradient descent algorithm satisfies*

$$\begin{aligned}
\overline{w} &= \sum_{i\in\mathcal{I}} \mu_i G_i(\overline{V}x_i + \overline{b}) - \sum_{j\in\mathcal{J}} \mu_j G_j(\overline{V}x_j + \overline{b}) \\
\overline{V} &= \sum_{i\in\mathcal{I}} \mu_i G_i \overline{w} x_i^\top - \sum_{j\in\mathcal{J}} \mu_j G_j \overline{w} x_j^\top \\
\overline{b} &= \sum_{i\in\mathcal{I}} \mu_i G_i \overline{w} - \sum_{j\in\mathcal{J}} \mu_j G_j \overline{w}
\end{aligned}$$

for some set of nonnegative scalars $\{\mu_s : s \in \mathcal{I}\cup\mathcal{J}\}$.

Proof Given the dynamics (24) of the gradient descent algorithm, define the set of support vectors as

$$\mathcal{I}^{\text{sup}} = \left\{ i\in\mathcal{I} : \lim_{t\to\infty} \frac{e^{-w^\top(Vx_{i'}+b)_+}}{e^{-w^\top(Vx_i+b)_+}} < \infty \;\forall i'\in\mathcal{I},\; \lim_{t\to\infty} \frac{e^{w^\top(Vx_{j'}+b)_+}}{e^{-w^\top(Vx_i+b)_+}} < \infty \;\forall j'\in\mathcal{J} \right\},$$

$$\mathcal{J}^{\text{sup}} = \left\{ j\in\mathcal{J} : \lim_{t\to\infty} \frac{e^{-w^\top(Vx_{i'}+b)_+}}{e^{w^\top(Vx_j+b)_+}} < \infty \;\forall i'\in\mathcal{I},\; \lim_{t\to\infty} \frac{e^{w^\top(Vx_{j'}+b)_+}}{e^{w^\top(Vx_j+b)_+}} < \infty \;\forall j'\in\mathcal{J} \right\}.$$

Note that the points corresponding to these indices dominate the dynamics of the algorithm as the training continues. Let $\overline{b}_{s_1}, \overline{b}_{s_2}$ be two nonzero coordinates of $\overline{b}$, let x_S be any of the support vectors. Then,

$$\begin{aligned}
\frac{\overline{b}_{s_1}}{\overline{b}_{s_2}} &= \lim_{t\to\infty} \frac{b_{s_1}(t)}{b_{s_2}(t)} \\
&= \lim_{t\to\infty} \frac{\frac{db_{s_1}}{dt}}{\frac{db_{s_2}}{dt}} \\
&= \lim_{t\to\infty} \frac{\sum_{i\in\mathcal{I}} e_{s_1}^\top G_i w \frac{e^{-w^\top G_i(Vx_i+b)}}{1+e^{-w^\top G_i(Vx_i+b)}} - \sum_{j\in\mathcal{J}} e_{s_1}^\top G_j w \frac{e^{w^\top G_j(Vx_j+b)}}{1+e^{w^\top G_j(Vx_j+b)}}}{\sum_{i\in\mathcal{I}} e_{s_2}^\top G_i w \frac{e^{-w^\top G_i(Vx_i+b)}}{1+e^{-w^\top G_i(Vx_i+b)}} - \sum_{j\in\mathcal{J}} e_{s_2}^\top G_j w \frac{e^{w^\top G_j(Vx_j+b)}}{1+e^{w^\top G_j(Vx_j+b)}}} \\
&= \lim_{t\to\infty} \frac{\sum_{i\in\mathcal{I}} e_{s_1}^\top G_i w \frac{e^{-w^\top G_i(Vx_i+b)}}{e^{w^\top G_S(Vx_S+b)}} - \sum_{j\in\mathcal{J}} e_{s_1}^\top G_j w \frac{e^{w^\top G_j(Vx_j+b)}}{e^{w^\top G_S(Vx_S+b)}}}{\sum_{i\in\mathcal{I}} e_{s_2}^\top G_i w \frac{e^{-w^\top G_i(Vx_i+b)}}{e^{w^\top G_S(Vx_S+b)}} - \sum_{j\in\mathcal{J}} e_{s_2}^\top G_j w \frac{e^{w^\top G_j(Vx_j+b)}}{e^{w^\top G_S(Vx_S+b)}}} \\
&= \frac{\sum_{i\in\mathcal{I}} e_{s_1}^\top G_i \overline{w} \mu_i - \sum_{j\in\mathcal{J}} e_{s_1}^\top G_j \overline{w} \mu_j}{\sum_{i\in\mathcal{I}} e_{s_2}^\top G_i \overline{w} \mu_i - \sum_{j\in\mathcal{J}} e_{s_2}^\top G_j \overline{w} \mu_j}
\end{aligned}$$

where

$$\begin{aligned}
\mu_i &\propto \lim_{t\to\infty} \frac{e^{-w^\top G_i(Vx_i+b)}}{e^{w^\top G_S(Vx_S+b)}} \quad \forall i\in\mathcal{I}, \\
\mu_j &\propto \lim_{t\to\infty} \frac{e^{-w^\top G_j(Vx_j+b)}}{e^{w^\top G_S(Vx_S+b)}} \quad \forall j\in\mathcal{J}.
\end{aligned}$$

Similarly, if $e_{s_1}^\top \overline{b}$ and $e_{s_2}^\top \overline{V} e_{s_3}$ are two nonzero elements of $\overline{b}$ and $\overline{V}$, then

$$\begin{aligned}
\frac{e_{s_1}^\top \overline{b}}{e_{s_2}^\top \overline{V} e_{s_3}} &= \lim_{t\to\infty} \frac{e_{s_1}^\top b}{e_{s_2}^\top V e_{s_3}} \\
&= \lim_{t\to\infty} \frac{e_{s_1}^\top \frac{db}{dt}}{e_{s_2}^\top \frac{dV}{dt} e_{s_3}} \\
&= \lim_{t\to\infty} \frac{\sum_{i\in\mathcal{I}} e_{s_1}^\top G_i w \frac{e^{-w^\top G_i(Vx_i+b)}}{e^{w^\top G_S(Vx_S+b)}} - \sum_{j\in\mathcal{J}} e_{s_1}^\top G_j w \frac{e^{w^\top G_j(Vx_j+b)}}{e^{w^\top G_S(Vx_S+b)}}}{\sum_{i\in\mathcal{I}} e_{s_2}^\top G_i w x_i e_{s_3} \frac{e^{-w^\top G_i(Vx_i+b)}}{e^{w^\top G_S(Vx_S+b)}} - \sum_{j\in\mathcal{J}} e_{s_2}^\top G_j w x_j e_{s_3} \frac{e^{w^\top G_j(Vx_j+b)}}{e^{w^\top G_S(Vx_S+b)}}} \\
&= \frac{\sum_{i\in\mathcal{I}} e_{s_1}^\top G_i \overline{w} \mu_i - \sum_{j\in\mathcal{J}} e_{s_1}^\top G_j \overline{w} \mu_j}{\sum_{i\in\mathcal{I}} e_{s_2}^\top G_i \overline{w} x_i e_{s_3} \mu_i - \sum_{j\in\mathcal{J}} e_{s_2}^\top G_j \overline{w} x_j e_{s_3} \mu_j}.
\end{aligned}$$

As a result, we have

$$\begin{aligned}
\overline{V} &= \sum_{i\in\mathcal{I}} \mu_i G_i \overline{w} x_i^\top - \sum_{j\in\mathcal{J}} \mu_j G_j \overline{w} x_j^\top \\
\overline{b} &= \sum_{i\in\mathcal{I}} \mu_i G_i \overline{w} - \sum_{j\in\mathcal{J}} \mu_j G_j \overline{w}
\end{aligned}$$

for some set of nonnegative scalars $\{\mu_s : s \in \mathcal{I} \cup \mathcal{J}\}$.

An identical analysis of $\overline{w}_{s_1}/\overline{w}_{s_2}$ for two nonzero coordinates of $\overline{w}$ shows that

$$\overline{w} = \sum_{i\in\mathcal{I}} \alpha\mu_i G_i(\overline{V}x_i + \overline{b}) - \sum_{j\in\mathcal{J}} \alpha\mu_j G_j(\overline{V}x_j + \overline{b})$$

for some $\alpha \in (0, \infty)$. In order to find the value of α, note that

$$\begin{aligned}
\langle \overline{w}, \overline{w} \rangle &= \left\langle \overline{w}, \sum_{i\in\mathcal{I}} \alpha\mu_i G_i(\overline{V}x_i + \overline{b}) - \sum_{j\in\mathcal{J}} \alpha\mu_j G_j(\overline{V}x_j + \overline{b}) \right\rangle \\
&= \alpha \left\langle \sum_{i\in\mathcal{I}} \mu_i G_i \overline{w} x_i^\top - \sum_{j\in\mathcal{J}} \mu_j G_j \overline{w} x_j^\top, \overline{V} \right\rangle + \alpha \left\langle \sum_{i\in\mathcal{I}} \mu_i G_i \overline{w} - \sum_{j\in\mathcal{J}} \mu_j G_j \overline{w}, \overline{b} \right\rangle \\
&= \alpha\langle \overline{V}, \overline{V} \rangle + \alpha\langle \overline{b}, \overline{b} \rangle
\end{aligned}$$

which shows that α must be 1 due to Lemma 5. This completes the proof. ■

Proof of Theorem 3. For any $k \in [r]$, let $\overline{w}_k$, $\overline{V}_k$ and $\overline{b}_k$ denote the k-th row of $\overline{w}$, $\overline{V}$ and $\overline{b}$, respectively. From Lemma 5, we have that

$$\overline{w}_k^2 = \|\overline{V}_k\|_2^2 + \overline{b}_k^2.$$

Let $\mathcal{I}^k \subseteq \mathcal{I}$ and $\mathcal{J}^k \subseteq \mathcal{J}$ denote the support vectors that activate the j-th node in the hidden layer, i.e.,

$$\begin{aligned}
\mathcal{I}^k &= \left\{ i \in \mathcal{I} : \overline{V}_k x_i + \overline{b}_k > 0,\ \overline{w}^\top (\overline{V}x_i + \overline{b})_+ = 1 \right\}, \\
\mathcal{J}^k &= \left\{ j \in \mathcal{J} : \overline{V}_k x_j + \overline{b}_k > 0,\ \overline{w}^\top (\overline{V}x_j + \overline{b})_+ = -1 \right\}.
\end{aligned}$$

Then we have

$$\begin{aligned}\overline{w}_k &= \sum\nolimits_{i\in\mathcal{I}^k} \mu_i(\overline{V}_k x_i + \bar{b}_k) - \sum\nolimits_{j\in\mathcal{J}^k} \mu_j(\overline{V}_k x_j + \bar{b}_k)\\ \overline{V}_k &= \sum\nolimits_{i\in\mathcal{I}^k} \mu_i \overline{w}_k x_i^\top - \sum\nolimits_{j\in\mathcal{J}^k} \mu_j \overline{w}_k x_j^\top\\ \bar{b}_k &= \sum\nolimits_{i\in\mathcal{I}^k} \mu_i \overline{w}_k - \sum\nolimits_{j\in\mathcal{J}^k} \mu_j \overline{w}_k.\end{aligned}$$

Plugging the expressions for $\overline{V}_k$ and $\bar{b}_k$ into that of $\overline{w}_k$:

$$\overline{w}_k = \overline{w}_k \mu^\top X_k^\top X_k \mu + \frac{1}{\overline{w}_k}\bar{b}_k^2,$$

which gives

$$\begin{aligned}\mu^\top X_k^\top X_k \mu = 1 - \frac{\bar{b}_k^2}{\overline{w}_k^2} &\implies \lambda_{\min}(X_k^\top X_k)\|\mu\|_2^2 \le 1 - \frac{\bar{b}_k^2}{\overline{w}_k^2},\\ &\implies \lambda_{\min}(X_k^\top X_k)\frac{\|\mu\|_1^2}{n_k} \le 1 - \frac{\bar{b}_k^2}{\overline{w}_k^2},\end{aligned}$$

where X_k is a matrix with columns $\{x_i : i \in \mathcal{I}^k\}$ and $\{-x_j : j \in \mathcal{J}^k\}$, μ is a column vector with elements $\{\mu_s : s \in \mathcal{I}^k \cup \mathcal{J}^k\}$, and n_k is the number of support vectors that activate node k. Note that if the columns of X_k are not linearly independent, then μ_s can be arbitrarily large for any support vector x_s. From the last inequality, we have

$$\sum\nolimits_{s\in\mathcal{I}^k\cup\mathcal{J}^k} \mu_s \le \frac{\sqrt{|\mathcal{I}^k| + |\mathcal{J}^k|}}{\sqrt{\lambda_{\min}(X_k^\top X_k)}}.$$

From the definition of the support vectors, we also have that

$$\begin{aligned}\sum\nolimits_{k\in[r]} \overline{w}_k^2 &= \sum\nolimits_{k\in[r]} \Big(\sum\nolimits_{i\in\mathcal{I}^k} \mu_i \overline{w}_k(\overline{V}_k x_i + \bar{b}_k) - \sum\nolimits_{j\in\mathcal{J}^k} \mu_j \overline{w}_k(\overline{V}_k x_j + \bar{b}_k)\Big)\\ &= \sum\nolimits_{s\in\mathcal{I}\cup\mathcal{J}} \mu_s\end{aligned}$$

due to complementary slackness condition.

Our goal is to bound the Lipschitz constant of the estimate $\hat{f}(x) = \overline{w}^\top(\overline{V}x + \bar{b})_+$:

$$\begin{aligned}\sum\nolimits_{k\in[r]} |\overline{w}_k|\|\overline{V}_k\|_2 &\le \sum\nolimits_{k\in[r]} \overline{w}_k^2\\ &= \sum\nolimits_{s\in\mathcal{I}\cup\mathcal{J}} \mu_s\\ &\le \frac{n_{\text{node}}^{\min}\sqrt{n_{\text{sup}}^{\max}}}{\min_{k\in[r]} \sqrt{\lambda_{\min}(X_k^\top X_k)}}\end{aligned}$$

where $n_{\text{node}}^{\min}$ denotes the minimum number of nodes that are activated by all of the support vectors:

$$n_{\text{node}}^{\min} = \min_{K\subseteq[r]} \left\{|K| : \forall s \in \mathcal{I}^{\text{sup}} \cup \mathcal{J}^{\text{sup}}\ \exists k \in K \text{ such that } \overline{V}_k x_s + \bar{b}_k > 0\right\}.$$

■

Appendix D. Proof of Theorem 4

The first order stationarity condition implies that

$$\frac{\partial \ell}{\partial W_k} = 0 \quad \forall k \in [L],$$

which yields

$$\begin{aligned} 0 &= \sum\nolimits_{s\in\mathcal{I}\cup\mathcal{J}} \overline{W}_{k+1}^\top \cdots \overline{W}_L^\top \left.\frac{\partial d(z,y_s)}{\partial z}\right|_{z=\overline{W}_L\cdots\overline{W}_1 x_s} x_s^\top \overline{W}_1^\top \cdots \overline{W}_{k-1}^\top + 2\mu_k \overline{W}_k \\ &= \overline{W}_{k+1}^\top \cdots \overline{W}_L^\top \left(\sum\nolimits_{s\in\mathcal{I}\cup\mathcal{J}} \left.\frac{\partial d(z,y_s)}{\partial z}\right|_{z=\overline{W}_L\cdots\overline{W}_1 x_s} x_s^\top\right) \overline{W}_1^\top \cdots \overline{W}_{k-1}^\top + 2\mu_k \overline{W}_k. \end{aligned}$$

Since the first term has rank 1, so does $\overline{W}_k$, for each $k \in [L]$. ■

Appendix E. Proof of Theorem 5

For brevity in notation, define

$$\begin{aligned} \tilde{x}_i &= x_i \quad\quad \forall i \in \mathcal{I}, \\ \tilde{x}_j &= -x_j \quad \forall j \in \mathcal{J}, \end{aligned}$$

and $\mathcal{S} = \mathcal{I} \cup \mathcal{J}$. Then the cost function (8) could be written as

$$\ell(W_1, \dots, W_L) = \sum\nolimits_{s\in\mathcal{S}} \log\left(1 + e^{W_L \cdots W_1 \tilde{x}_s}\right).$$

Note that at least one of the weight matrices must diverge to ∞ in norm since the loss function does not attain its minimum at a finite point. In addition, we have

$$\begin{aligned} \frac{d\|W_k(t)\|_F^2}{dt} &= 2\left\langle W_k(t), \frac{dW_k(t)}{dt}\right\rangle \\ &= 2\left\langle W_k(t), -\frac{\partial \ell}{\partial W_k}\right\rangle \\ &= \frac{d\|W_{k'}(t)\|_F^2}{dt} \quad \forall k, k' \in [L], \end{aligned}$$

and consequently,

$$\|W_k(t)\|_F^2 - \|W_{k'}(t)\|_F^2 = \|W_k(0)\|_F^2 - \|W_{k'}(0)\|_F^2 \quad \forall t \geq 0,\ \forall k, k' \in [L].$$

Therefore, all of the weight matrices must diverge to ∞ in norm.

We can define the set of indices for the support vectors:

$$\mathcal{S}^{\text{sup}} = \left\{ s \in \mathcal{S} : \lim_{t\to\infty} \frac{\exp(W_L \cdots W_1 \tilde{x}_{s'})}{\exp(W_L \cdots W_1 \tilde{x}_s)} < \infty\ \forall s' \in \mathcal{S} \right\},$$

which is a nonempty set. For any $k \in [L]$, remember that $\overline{W}_k \in \mathbb{R}^{n_k \times n_{k-1}}$. Let $i_1, i_2 \in [n_k]$ and $j_1, j_2 \in [n_{k-1}]$ be such that $e_{i_1}^\top \overline{W}_k e_{j_1}$, $e_{i_2}^\top \overline{W}_k e_{j_1}$ and $e_{i_2} \overline{W}_k e_{j_2}$ are nonzero. Note that if such a tuple of (i_1, i_2, j_1) does not exist, then the matrix $\overline{W}_k$ has rank 1 already, and the following analysis is not needed. Given such a tuple of (i_1, i_2, j_1), we have

$$\begin{aligned}
\frac{e_{i_1}^\top \overline{W}_k e_{j_1}}{e_{i_2}^\top \overline{W}_k e_{j_1}} &= \lim_{t\to\infty} \frac{e_{i_1}^\top W_k(t) e_{j_1}}{e_{i_2}^\top W_k(t) e_{j_1}} \\
&= \lim_{t\to\infty} \frac{e_{i_1}^\top \frac{dW_k}{dt} e_{j_1}}{e_{i_2}^\top \frac{dW_k}{dt} e_{j_1}} \\
&= \lim_{t\to\infty} \frac{\sum_{s\in\mathcal{S}} e_{i_1}^\top \left(W_{k+1}^\top \cdots W_L^\top \tilde{x}_s^\top W_1^\top \cdots W_{k-1}^\top\right) e_{j_1} e^{W_L \cdots W_1 \tilde{x}_s}}{\sum_{s\in\mathcal{S}} e_{i_2}^\top \left(W_{k+1}^\top \cdots W_L^\top \tilde{x}_s^\top W_1^\top \cdots W_{k-1}^\top\right) e_{j_1} e^{W_L \cdots W_1 \tilde{x}_s}} \\
&= \frac{e_{i_1}^\top \overline{W}_{k+1}^\top \cdots \overline{W}_L^\top}{e_{i_2}^\top \overline{W}_{k+1}^\top \cdots \overline{W}_L^\top} \lim_{t\to\infty} \frac{\sum_{s\in\mathcal{S}^{\text{sup}}} \left(\tilde{x}_s^\top \overline{W}_1^\top \cdots \overline{W}_{k-1}^\top\right) e_{j_1} e^{W_L \cdots W_1 \tilde{x}_s - W_L \cdots W_1 \tilde{x}_{s_{\text{sup}}}}}{\sum_{s\in\mathcal{S}^{\text{sup}}} \left(\tilde{x}_s^\top \overline{W}_1^\top \cdots \overline{W}_{k-1}^\top\right) e_{j_1} e^{W_L \cdots W_1 \tilde{x}_s - W_L \cdots W_1 \tilde{x}_{s_{\text{sup}}}}} \\
&= \frac{e_{i_1}^\top \overline{W}_{k+1}^\top \cdots \overline{W}_L^\top}{e_{i_2}^\top \overline{W}_{k+1}^\top \cdots \overline{W}_L^\top} \lim_{t\to\infty} \frac{\sum_{s\in\mathcal{S}^{\text{sup}}} \left(\tilde{x}_s^\top \overline{W}_1^\top \cdots \overline{W}_{k-1}^\top\right) e_{j_2} e^{W_L \cdots W_1 \tilde{x}_s - W_L \cdots W_1 \tilde{x}_{s_{\text{sup}}}}}{\sum_{s\in\mathcal{S}^{\text{sup}}} \left(\tilde{x}_s^\top \overline{W}_1^\top \cdots \overline{W}_{k-1}^\top\right) e_{j_2} e^{W_L \cdots W_1 \tilde{x}_s - W_L \cdots W_1 \tilde{x}_{s_{\text{sup}}}}} \\
&= \lim_{t\to\infty} \frac{e_{i_1}^\top \frac{dW_k}{dt} e_{j_2}}{e_{i_2}^\top \frac{dW_k}{dt} e_{j_2}} \\
&= \frac{e_{i_1}^\top \overline{W}_k e_{j_2}}{e_{i_2}^\top \overline{W}_k e_{j_2}}
\end{aligned}$$

where $\tilde{x}_{s_{\text{sup}}}$ denotes any point in the set $\{x_s\}_{s\in\mathcal{S}^{\text{sup}}}$. Note that this equality shows that all columns of $\overline{W}_k$ are in the span of the same single vector in $\mathbb{R}^{n_k}$, which proves that $\overline{W}_k$ has rank 1. ■

Appendix F. Proof of Theorem 6

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

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

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

Then problem (12b) 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:

$$\begin{aligned} \underset{w,t}{\text{minimize}} \quad & \sum\nolimits_{i\in\mathcal{I}} \left(y_i - w^\top x_i\right)^2 + \epsilon^2 t \\ \text{subject to} \quad & \|w\|_p^2 \le t. \end{aligned} \tag{25}$$

Fix $\epsilon > 0$, and assume that (w_0, t_0) is the solution of (25). Then w_0 is also a solution to the problem

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

as well as

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \sum\nolimits_{i\in\mathcal{I}} (y_i - w^\top x_i)^2 \\ \text{subject to} \quad & \|w\|_p^m \le t_0^{m/2}. \end{aligned} \tag{26}$$

If $t_0 = 0$, then w_0 is also zero, and this solution can be obtained by (12a) by choosing λ large enough. Therefore, without loss of generality assume $t_0 \neq 0$. Then problem (26) satisfies the Slater's condition, and strong duality holds (Boyd and 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

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

This completes the one direction of the proof, and the other direction is identical. ■