J. He et al. *Res Math Sci(2023)10:13*
https://doi.org/10.1007/s40687-023-00378-y

Research in the Mathematical Sciences

RESEARCH

Side effects of learning from low-dimensional data embedded in a Euclidean space

Juncai He[1*], Richard Tsai[2] and Rachel Ward[2]

*Correspondence:
jhemath@outlook.com
[1]Department of Mathematics, The University of Texas at Austin, Austin, TX 78712, USA
Full list of author information is available at the end of the article

Abstract

The low-dimensional manifold hypothesis posits that the data found in many applications, such as those involving natural images, lie (approximately) on low-dimensional manifolds embedded in a high-dimensional Euclidean space. In this setting, a typical neural network defines a function that takes a finite number of vectors in the embedding space as input. However, one often needs to consider evaluating the optimized network at points outside the training distribution. This paper considers the case in which the training data are distributed in a linear subspace of $\mathbb{R}^d$. We derive estimates on the variation of the learning function, defined by a neural network, in the direction transversal to the subspace. We study the potential regularization effects associated with the network's depth and noise in the codimension of the data manifold. We also present additional side effects in training due to the presence of noise.

Keywords: Low-dimensional data, Learning, Neural networks, Side effects, Regularization

Mathematics Subject Classification: 68P01, 68T07

1 Introduction

In many machine learning problems, one observes that data points typically concentrate on a lower-dimensional manifold embedded in $\mathbb{R}^d$. Indeed, the low-dimensional manifold hypothesis [21,29,40,42,53] posits that the data found in many applications, such as those involving natural images, lie (approximately) on low-dimensional manifolds which are embedded in high-dimensional coding spaces. Manifold learning algorithms [8,20,44,45,53,55] aim at finding low-dimensional representations of the high-dimensional data. There are many supervised or unsupervised linear dimensionality reduction methods. We mention linear discriminant analysis (LDA) [6], principal component analysis (PCA) [1], multiple dimensional scaling (MDS) [18], and canonical correlation analysis (CCA) [26]. The random projection framework for data compression provides a theoretical framework for justification [10,31,34]. Nevertheless, even after a suitable dimension reduction, it is common to find that the data still concentrate on some lower-dimensional manifold embedded in a higher-dimensional Euclidean space. This is at odds with the typical (and

© The Author(s), under exclusive licence to Springer Nature Switzerland AG 2023.

crucial) assumption found in many supervised machine learning theories: that the labeled data points are drawn i.i.d. from a probability distribution whose support has full measure in the embedding space [9].

In this paper, we will assume that the data points are sampled from a linear subspace $\mathcal{M}$ of $\mathbb{R}^d$ and take the form $(\boldsymbol{x}, g(\boldsymbol{x})) \in \mathbb{R}^d \times \mathbb{R}$, where $\boldsymbol{x} \in \mathcal{M}$, $\dim(\mathcal{M}) < d$, and $g : \mathcal{M} \mapsto \mathbb{R}$ is a smooth function. The data points are used to identify a function $f_{\theta^*} : \mathbb{R}^d \mapsto \mathbb{R}$ from a parameterized family of functions f_θ defined by particular neural network architecture. The "trained" function f_{θ^*} is constructed by optimizing the network's parameters θ to fit the given data. The approximation properties of neural networks for functions defined on embedded low-dimensional manifolds are studied in [12,16,17,36,47,48]. However, due to the presence of noise, the limitation to the training data acquisition, or distribution shift in the data that occurs post-training, one often needs to evaluate f_{θ^*} on points in a manifold $\mathcal{M}'$ which is close to but not identical to $\mathcal{M}$. As such, the behavior of the trained neural network f_{θ^*} on $\mathcal{M}'$ is a nontrivial but practically important question. Not surprisingly, the performance of the trained network f_{θ^*} off of the data manifold $\mathcal{M}$ is more consistent the less that f_{θ^*} varies in the normal direction of $\mathcal{M}$. This becomes a question of estimating the magnitude of $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$, with $n_{\mathcal{M}}$ denoting a normal direction of $\mathcal{M}$. These observations motivate the following questions: Can $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$ be regulated by choice of neural network architecture and optimization method? In which ways can noisy training data improve the stability performance of learning a neural network with low-dimensional data? How does the low-dimensional structure of the data manifold affect the stability of the performance of the trained neural network when applied to points away from the data manifold?

We will analyze the training process of f_θ and the properties of $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$ for deep linear neural networks or a nonlinear networks activated by ReLU. We aim to reveal the effect of the arbitrariness of ambient space on the optimized neural networks. We will also discuss the approach of introducing noise to the non-label components of training data for reducing the effect of this "arbitrariness," i.e., for the regulation of $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$. In many applications, principal component analysis can be used to reveal the low-dimensional aspects of the data set. In those cases, the data sets can be described as samples from distributions with specific variances from a sequence of linear subspaces in a Euclidean ambient space. The analysis in this paper is highly relevant.

The main contributions of this paper are listed below:

1. If the data points, including noise, lie on $\mathcal{M}$, the linear network's depth may provide certain implicit regularization or side effects as shown in Fig. 7 and Theorem 4. For ReLU neural networks, Theorems 5, 6, and Corollary 2 show that $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$ is sensitive to the initialization of a set of "untrainable" parameters.
2. If the noise has a small positive variance in the orthogonal complement of $\mathcal{M}$, then:
   - $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$ can be made arbitrarily small, provided that the number of data points scales according to some inverse power of the variance as shown in Theorem 1 for deep linear neural networks and Fig. 12 for deep nonlinear neural networks. From our experiments, the scaling laws for nonlinear ReLU networks are significantly different from the linear networks—much more data points are needed to control the size of $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$;

– We show that gradient descent algorithms can be very inefficient. The time needed for the gradient descent dynamics to reach a small neighborhood of the optimal parameters is reciprocal of the data set's variance in the normal space of $\mathcal{M}$. See Theorem 2. In addition, it may also need a long time to escape the near region of origin as shown in Theorem 3.

3. The stability–accuracy trade-off. The role of noise can be interpreted as a stabilizer for a model when evaluated on points outside of the (clean) data distribution. The regularization effect is equivalent to changing the loss function for learning functions defined in the ambient space. However, adding noise to the data set will impact of the accuracy of the network's generalization error (for evaluation within the data distribution). For nonlinear data manifolds, uniform noise may render the labeled data incompatible.

In the remainder of this section, we define the basic setting that we will work with and discuss the linear regression problem under this settings to motivate the rest of the paper. In Sect. 2, we present some special challenges in training deep linear neural networks via gradient descent. These challenges arise from embedding of data in a higher-dimensional space. We will derive estimates for stability for linear networks in Sect. 2 and nonlinear networks activated by ReLU in Sect. 3. In Sect. 4, we briefly discuss the regularization of $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$ by adding noise to data globally and the stability–accuracy trade-off. In Sect. 5, we give a final summary.

1.1 The basic setting

Let $\mathcal{M}$ be a lower-dimensional subspace of $\mathbb{R}^d$ defined as follows:

$$\mathcal{M} = \left\{ \boldsymbol{x} = Q \begin{pmatrix} x \\ 0 \end{pmatrix} \in \mathbb{R}^d : x \in \mathbb{R}^{d_x} \right\}$$

with Q representing a unitary matrix, here and throughout. Consider the distribution of points in $\mathbb{R}^d$ following

$$M_\sigma := Q \begin{pmatrix} X \\ \sigma Y \end{pmatrix},$$

where $\sigma \geq 0$, $Q \in \mathbb{R}^{d\times d}$ is a unitary matrix, and $X \in \mathbb{R}^{d_x}$ is a random vector representing the underlying distribution of data and $Y \in \mathbb{R}^{d_y}$ is a random vector independent from X. Y is assumed to sample either the normal distribution $N(0, I_{d_y})$ or the uniform distribution $U\left([-1,1]^{d_y}\right)$. Y represents the noise model in the dimensions normal to $\mathcal{M}$. In particular, $\boldsymbol{x} \in \mathcal{M}$ if $\boldsymbol{x}$ is sampled from M_0. Finally, we consider labeled training data of the form

$$D_N := \{(\boldsymbol{x}_i, g_i)\}_{i=1}^N, \quad \boldsymbol{x}_i \sim M_\sigma, g_i \in \mathbb{R}, \tag{1.1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ is of the form

$$\boldsymbol{x}_i = Q \begin{pmatrix} x_i \\ \sigma y_i \end{pmatrix} \in \mathbb{R}^{d_x+d_y}, \quad \sigma \geq 0, \tag{1.2}$$

with $x_i \sim X$, $y_i \sim Y$, and $d = d_x + d_y$. We further assume that

$$\operatorname{rank}\left(\sum_{i=1}^N x_i x_i^T\right) = d_x, \tag{1.3}$$

or equivalently, that the matrix $(x_1|x_2|\cdots|x_N)$ has full rank. This means that the data do samples every subspace of $\mathcal{M}$.

A crucial assumption in our paper is that the target function only depends on x_i, i.e., there exists a function $g : \mathbb{R}^{d_x} \mapsto \mathbb{R}$ such that

$$g_i = g(x_i) \in \mathbb{R}.$$

However, we point out that the typical learning model and training algorithms are agnostic to this assumption. As a result, we design our machine learning model $f_\theta : \mathbb{R}^d \mapsto \mathbb{R}$ rather than $\mathbb{R}^{d_x} \mapsto \mathbb{R}$.

A typical machine learning model with parameter set $\theta \in \mathbb{R}^p$ is used to define a function

$$f_\theta(\cdot) = f(\cdot;\theta) : \mathbb{R}^d \mapsto \mathbb{R}.$$

In particular, we study the case of $f(\boldsymbol{x};\theta)$ being a deep neural network

$$\begin{cases} f^\ell(\boldsymbol{x}) & = W^\ell \alpha(f^{\ell-1}(\boldsymbol{x})) + b^\ell, \quad \ell = 2 : L, \\ f(\boldsymbol{x};\theta) & = f^L(\boldsymbol{x}), \end{cases} \tag{1.4}$$

where $f^1(\boldsymbol{x}) = W^1\boldsymbol{x} + b^1$, $W^\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$, and $b^\ell, f^\ell \in \mathbb{R}^{n_\ell}$ with $n_0 = d$ and $n_L = 1$. Here, $\theta = \{(W^\ell, b^\ell)\}_{\ell=1}^L$ denotes the set of all parameters in the deep neural network $f(\boldsymbol{x};\theta)$. In the following, we will focus on two different networks:

1. linear networks:

$$\alpha(x) = x \quad \text{and} \quad b^\ell \equiv 0; \tag{1.5}$$

2. ReLU-activated neural networks:

$$\alpha(x) = \mathrm{ReLU}(x) := \max\{0, x\}. \tag{1.6}$$

A trained function f_{θ^*} is constructed by gradient descent applied to the optimization problem

$$\min_{\theta\in\mathbb{R}^p} J(\theta), \qquad J(\theta) = \frac{1}{2N}\sum_{i=1}^N |f_\theta(\boldsymbol{x}_i) - g_i|^2. \tag{1.7}$$

More precisely, θ is updated by first initializing as θ^0 and then updating

$$\theta^{t+1} = \theta^t - \eta_t \frac{\partial J(\theta^t)}{\partial \theta} \tag{1.8}$$

with some $\eta_t > 0$ for $t \geq 0$. In this paper, we shall refer to this updating scheme as (full) gradient descent (FGD). We will also discuss the typical stochastic gradient descent (SGD) update, where J and $\frac{\partial J}{\partial \theta}$ are replaced, respectively, by J_{B_t} and $\frac{\partial J_{B_t}}{\partial \theta}$, and

$$J_{B_t}(\theta) = \sum_{\boldsymbol{x}_i \in B_t} |f_\theta(\boldsymbol{x}_i) - g_i|^2,$$

where $B_t \subsetneq \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ is randomly chosen and called a mini-batch.

Let

$$\mathcal{P}_{\mathcal{M}}\boldsymbol{x} := Q \begin{pmatrix} I_{d_x} & 0 \\ 0 & 0 \end{pmatrix} Q^T \boldsymbol{x},$$

where I_{d_x} is the $d_x \times d_x$ identity matrix, and define $\overline{g} : \mathbb{R}^d \mapsto \mathbb{R}$ as

$$\overline{g}(\boldsymbol{x}) = g\left(\mathcal{P}_{\mathcal{M}}\boldsymbol{x}\right). \tag{1.9}$$

$\mathcal{P}_{\mathcal{M}}$ is the orthogonal projection onto $\mathcal{M}$, and $\overline{g}(\boldsymbol{x})$ is the extension of $g(x)$ that stays constant in the directions orthogonal to $\mathcal{M}$. Correspondingly, we define $\overline{f}_\theta$ as the restriction of f_θ on $\mathcal{M}$:

$$\overline{f}_\theta(\boldsymbol{x}) = f_\theta(\mathcal{P}_{\mathcal{M}}\boldsymbol{x}).$$

Now consider $\mathcal{M}'$, which is close to but not necessarily identical to $\mathcal{M}$. We can estimate the error:

$$\left|f_{\theta^*}(\boldsymbol{x}) - \overline{g}(\boldsymbol{x})\right| \le \left|f_{\theta^*}(\boldsymbol{x}) - \overline{f}_{\theta^*}(\boldsymbol{x})\right| + \left|\overline{f}_{\theta^*}(\boldsymbol{x}) - \overline{g}(\boldsymbol{x})\right|, \quad \boldsymbol{x} \in \mathcal{M}', \tag{1.10}$$

where f_{θ^*} is learned from $\mathcal{M}_\sigma$ (clean data for $\sigma = 0$ or noisy data for $\sigma > 0$). The first term on the right-hand side can be interpreted as the stability error of the learned neural network $f_{\theta^*}(\boldsymbol{x})$. It measures the amount $f_{\theta^*}(\boldsymbol{x})$ that varies along the normal direction of the subspace $\mathcal{M}$. In particular, we have

$$\left|f_{\theta^*}(\boldsymbol{x}) - \overline{f}_{\theta^*}(\boldsymbol{x})\right| \le \left\| \frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}} \right\| \|\boldsymbol{x} - \mathcal{P}_{\mathcal{M}}\boldsymbol{x}\|, \quad \boldsymbol{x} \in \mathcal{M}'. \tag{1.11}$$

The term $\|\boldsymbol{x} - \mathcal{P}_{\mathcal{M}}\boldsymbol{x}\|$ is controlled by the difference between the data subspace $\mathcal{M}$ and the test set in $\mathcal{M}'$. The second term on the right-hand side of (1.10) corresponds to the approximation ability of the neural network. An approximation theory of neural networks for functions of the form $\overline{g}(\boldsymbol{x}) = g(\mathcal{P}_{\mathcal{M}}\boldsymbol{x})$ is established in [17], where $\mathcal{M}$ is a general manifold and $\mathcal{P}_{\mathcal{M}}\boldsymbol{x} = \arg\inf_{\boldsymbol{\xi}\in\mathcal{M}} \|\boldsymbol{x}-\boldsymbol{\xi}\|$ defines the orthogonal projection onto a general manifold $\mathcal{M}$. In other words, in [17] the data are assumed to be sampled from $(\boldsymbol{x}, \overline{g}(\boldsymbol{x}))$, where $\boldsymbol{x} \in \mathcal{A} \subset [0,1]^d$ and $\mathcal{A}$ is assumed to be contained in a tubular region around $\mathcal{M}$. Provided that the tubular region has a radius smaller than the reach of $\mathcal{M}$, $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$ of an optimal network would be 0 in the tubular region.

We remark that in the typical machine learning setup, one considers data sampled from the same manifold, which corresponds to $\mathcal{M}' \equiv \mathcal{M}$. In comparison, we are interested in deriving bounds for "out of distribution" error or a kind of stability metric. Thus, we shall focus on (1.11), the right-hand side of (1.10), and assume that the second term can be bounded appropriately.

In this paper, the empirical means of quantities derived from the data will often play a role. We adopt the following notation:

Notation 1 *Let z be a random variable in $\mathbb{R}^m$ or $\mathbb{R}^{m\times n}$ over some probability space and let z_i denote a sample realization of z. We denote the empirical average*

$$\langle z\rangle_N := \frac{1}{N}\sum_{i=1}^{N} z_i$$

and the mean

$$\langle z\rangle := \lim_{N\to\infty} \langle z\rangle_N = \mathbb{E}[z].$$

Notation 2 *For vectors $(x_i, y_i) \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, $i = 1, 2, \cdots, N$, we denote the averaged correlation matrix by*

$$\langle A(x,y)\rangle_N := \begin{pmatrix} \langle xx^T\rangle_N & \langle xy^T\rangle_N \\ \langle yx^T\rangle_N & \langle yy^T\rangle_N \end{pmatrix}.$$

Unless explicitly stated otherwise, we will refer to $\langle A(x,y)\rangle_N$ as $\langle A\rangle_N$, and $\langle A(x,\sigma y)\rangle_N$ as $\langle A_\sigma\rangle_N$.

1.2 Warm up: linear regression

As a special case of linear neural networks, we first use simple linear regression to demonstrate how $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$ can be affected by the data and the model. Since Q can be factored into parameters, without loss of generality, we will assume that $Q \equiv I$.

For linear regression, f_θ, with $\theta \equiv \boldsymbol{w} \in \mathbb{R}^d$, takes the form

$$f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x} = w_x^T x + w_y^T y, \tag{1.12}$$

where $w_x \in \mathbb{R}^{d_x}$ and $w_y \in \mathbb{R}^{d_y}$. We solve

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{2N} \sum_{i=1}^{N} \left(\boldsymbol{w}^T \boldsymbol{x}_i - g_i \right)^2, \tag{1.13}$$

where $\boldsymbol{x}_i \sim M_\sigma$.

If $\sigma = 0$, in which case $y \equiv 0$ equivalently, the loss defined in (1.13) reduces to

$$J(\boldsymbol{w}) = \frac{1}{2N} \sum_{i=1}^{N} \left(w_x^T x_i + w_y^T\, 0 - g_i \right)^2.$$

Every point in the set $\{(w_x^*, w_y) \mid w_y \in \mathbb{R}^{d_y}, w_x^* = \langle xx^T \rangle_N^{-1} \langle gx \rangle_N\}$ is a minimizer. However, if gradient descent is used for the minimization, the "optimal" model takes the form

$$f(\boldsymbol{x}; \boldsymbol{w}^*) = (\boldsymbol{w}^*)^T \boldsymbol{x} = (w_x^*)^T x + (w_y^{(0)})^T y,$$

where $w_y^{(0)}$ is the initial value set for the gradient descent since $\frac{\partial J(\boldsymbol{w})}{\partial w_y} = 0$. Hence, we have

$$\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}} = \frac{\partial f(\boldsymbol{x}; \boldsymbol{w}^*)}{\partial y} = w_y^{(0)},$$

where $w_y^{(0)}$ keeps its initialization value. This means $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$ is determined by the initialization of w_y and does not change during the training process.

In the case $\sigma \neq 0$ and $d_x = d_y = 1$, there is a unique minimizer (w_x^*, w_y^*) that can be quickly derived:

$$w_x^* = \frac{\langle gx \rangle_N \langle y^2 \rangle_N - \langle gy \rangle_N \langle xy \rangle_N}{\langle x^2 \rangle_N \langle y^2 \rangle_N - \langle xy \rangle_N^2}, \quad w_y^* = \frac{1}{\sigma} \frac{\langle gy \rangle_N \langle x^2 \rangle_N - \langle gx \rangle_N \langle xy \rangle_N}{\langle x^2 \rangle_N \langle y^2 \rangle_N - \langle xy \rangle_N^2}.$$

In addition, if we assume that the distribution of x_i and y_i is independent and $\mathbb{E}[xy] = 0$, then we will have $\langle xy \rangle_N \sim \mathcal{O}(1/\sqrt{N})$, $\langle x^2 \rangle_N = \langle y^2 \rangle_N \sim \mathcal{O}(1)$, $\langle xg \rangle_N \sim \mathcal{O}(1)$, and $\langle yg \rangle_N \sim \mathcal{O}(1/\sqrt{N})$. This makes the following estimates hold with high probability:

$$w_x^* = \frac{\langle gx \rangle_N}{\langle x^2 \rangle_N} + \mathcal{O}\left(\frac{1}{N} \right)$$

and

$$w_y^* = \frac{1}{\sigma\sqrt{N}} \frac{\langle x^2 \rangle_N - \langle xg \rangle_N}{\langle x^2 \rangle_N \langle y^2 \rangle_N - \mathcal{O}(1/N)} \sim \mathcal{O}\left(\frac{1}{\sigma\sqrt{N}} \right).$$

To have $w_y^* \sim \mathcal{O}(1)$ as $\sigma \to 0$, one needs to take N to infinity according to

$$N \sim \mathcal{O}(\sigma^{-2}). \tag{1.14}$$

In other words, the resulting linear function will have a small normal derivative only if the number of data points scales super linearly inversely with the variance of the noise in the codimensions of $\mathcal{M}$.

The linear regression example reveals an important aspect about learning from embedded low-dimensional data that are persistent in more general settings. $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$ depends on the set of parameters which are not trainable when there is no noise. The smaller $\frac{\partial f_{\theta^*}}{\partial n_{\mathcal{M}}}$ is, the more stable the network is for evaluation at points out of training data distribution. In the presence of noise with small variance in the codimension directions, the number of training examples needs to scale inversely proportional to the variance.

2 Linear neural networks

In this section, we study learning with deep linear multilayer neural networks, in particular the gradient descent dynamics for minimizing the mean squared error. Regression with multiple hidden layer linear networks generalizes simple linear regression models. The training of linear neural networks provides a way to construct linear operators satisfying certain structural constraints [4,33]. Consequently, LNN models can be adapted to improve the performance of classic methods, for example in wave propagation [41] and linear convolutional neural networks in multigrid [14,27,30].

As defined in (1.4) and (1.5) we have the linear network with $L-1$ hidden layers as

$$f(\boldsymbol{x};\theta) = W^L W^{L-1}\cdots W^2 W^1 \boldsymbol{x} = \boldsymbol{w}^T \boldsymbol{x}, \tag{2.1}$$

where $\theta = (W^1, W^2, \cdots, W^L)$ denotes all parameter matrices in this model and the end-to-end parameter $\boldsymbol{w} = W^L W^{L-1}\cdots W^2 W^1$ is defined as the product of the W^k matrices. Here, $W^k \in \mathbb{R}^{n_k \times n_{k-1}}$ are the weights connecting the $(k-1)$th and the kth layer, $k = 1, 2, \cdots, L$, with the convention that the 0th layer is the input layer ($n_0 = d$) and Lth layer is the output layer ($n_L = 1$). In particular, we consider only the fixed-width case, i.e., $n_k = n \geq d$ for all $k = 1, 2, \cdots, L-1$. We will refer to such networks as LNNs.

We denote the loss function in terms of $(W^1, \cdots, W^L)$ as

$$J(W^1, \cdots, W^L) = \frac{1}{2N}\sum_{i=1}^{N} |W^L W^{L-1}\cdots W^2 W^1 \boldsymbol{x}_i - g_i|^2, \tag{2.2}$$

and in terms of the end-to-end parameters $\boldsymbol{w}$ as

$$J^e(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_i - g_i)^2, \tag{2.3}$$

where $\boldsymbol{x}_i \sim M_\sigma$. Here, the superscript e in J^e emphasizes the fact that J^e is the corresponding loss function for the end-to-end weight set $\boldsymbol{w}$.

In [3], Arora et al. proposed to minimize $J(W^1, W^2, \cdots, W^L)$ in terms of $(W^1, \cdots, W^L)$, and derived that gradient descent of J via the explicit stepping

$$W^\ell \leftarrow W^\ell - \eta \frac{\partial J}{\partial W^\ell}, \quad \ell = 1, 2, \cdots, k,$$

leads to the following dynamical system for $\boldsymbol{w}$ in the limit of $\eta \to 0$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} = -\|\boldsymbol{w}\|^{2-\frac{2}{L}} \left(\nabla_{\boldsymbol{w}} J^e(\boldsymbol{w}) + (L-1)\mathcal{P}_{\boldsymbol{w}} \left(\nabla_{\boldsymbol{w}} J^e(\boldsymbol{w})\right)\right), \tag{2.4}$$

under the assumptions for the initialization of $(W^1, \cdots, W^L)$ that

$$\left(W^{\ell+1}\right)^T W^{\ell+1} = W^\ell \left(W^\ell\right)^T \tag{2.5}$$

for all $\ell = 1 : L-1$. Here, $\mathcal{P}_{\boldsymbol{w}}(\cdot)$ denotes the operator that projects vectors onto the subspace spanned by $\boldsymbol{w}$:

$$\mathcal{P}_{\boldsymbol{w}}(\boldsymbol{v}) = \frac{\boldsymbol{w}\boldsymbol{w}^T}{\|\boldsymbol{w}\|^2}\boldsymbol{v}.$$

For convenience, we define the vector field $\boldsymbol{F} : \mathbb{R}^d \mapsto \mathbb{R}^d$ as

$$\boldsymbol{F}(\boldsymbol{w}) := -\|\boldsymbol{w}\|^{2-\frac{2}{L}} \left(\nabla_{\boldsymbol{w}} J^e(\boldsymbol{w}) + (L-1)\mathcal{P}_{\boldsymbol{w}} \left(\nabla_{\boldsymbol{w}} J^e(\boldsymbol{w})\right)\right). \tag{2.6}$$

Prior works related to LNNs with full-rank data. Early work on LNNs focused more on the side effects of introducing more hidden layers. For example, the ℓ^2 regression with two hidden linear layers was studied in [22]. In that paper, the author studied the training process and demonstrated the existence of overtraining under the so-called over-realizable cases by employing the exact solution for a matrix Riccati equation. A simplified nonlinear dynamical system was introduced in [46] to show that increasing depth in linear neural networks may slow down the training. However, it was proven in [32] that every local minimum is a global minimum for over-parameterized LNNs (width n is larger than the number of data N). It is shown recently in [3] that involving more linear layers beyond the simplest linear regression brings some advantages to the training of networks and possibly to the network's generalization performance. It is also reported in [3] that (2.4) yields an accelerated convergence of $\boldsymbol{w}$ compared to the linear regression case. Recently, the convergence of gradient flows related to learning deep LNNs was further studied in [5,39] by re-interpreting them as Riemannian gradient flows on the manifold of rank-r matrices endowed with a suitable Riemannian metric. It is worth stressing again that all these convergence results are established based on the assumption that $\langle \boldsymbol{x}\boldsymbol{x}^T\rangle_N$ is full rank.

In the remainder of this section, we aim at analyzing (2.4) in the context of embedded low-dimensional data.

2.1 Gradient descent for deep linear neural networks

In this subsection, we first study some general properties of the dynamical system (2.4). Then, we provide some further results if we involve the low-dimensional assumption of data. We first point out that the dynamical system (2.4) is invariant under unitary transformation:

Proposition 1 *Suppose that the data* $\{(\boldsymbol{x}_i, g_i)\}_{i=1}^N$ *follows* $\boldsymbol{x}_i = Q\begin{pmatrix} x_i \\ \sigma y_i \end{pmatrix} \sim \mathcal{M}_\sigma$ *for some unitary transform* Q *on* $\mathbb{R}^d$. *Denote* $\widetilde{\boldsymbol{x}}_i = Q^T\boldsymbol{x}_i$ *and* $\widetilde{\boldsymbol{w}} = Q^T\boldsymbol{w}$. *If* $\boldsymbol{w}(t)$ *satisfies* (2.4) *then* $\widetilde{\boldsymbol{w}}(t)$ *also satisfies* (2.4), *and vice versa.*

Thus, without loss of generality, we can focus on the case of $Q = I_d$, that is, $\mathcal{M} = \mathrm{Span}\{e_1, e_2, \cdots, e_{d_x}\}$. In this setup,

$$J^e(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=1}^{N}(w_x^T x_i + w_y^T \sigma y_i - g_i)^2. \tag{2.7}$$

Next, we derive the gradient of the loss function J^e:

$$\nabla_{\boldsymbol{w}} J^e(\boldsymbol{w}) = \langle A_\sigma\rangle_N \boldsymbol{w} - \langle g\boldsymbol{x}\rangle_N, \tag{2.8}$$

where $\langle A_\sigma\rangle_N$ is defined in Notation 2 and $\langle g\boldsymbol{x}\rangle_N = \begin{pmatrix} \langle gx\rangle_N \\ \sigma\langle gy\rangle_N \end{pmatrix}$ by definition in Notation 1. Here, we notice the relation between $\langle A_\sigma\rangle_N$ and $\langle A\rangle_N$

$$\langle A_\sigma\rangle_N = \begin{pmatrix} I_{d_x} & 0 \\ 0 & \sigma I_{d_y} \end{pmatrix}\langle A\rangle_N \begin{pmatrix} I_{d_x} & 0 \\ 0 & \sigma I_{d_y} \end{pmatrix}, \tag{2.9}$$

which is useful in the following analysis.

Then, we summarize some observations about the stationary points of (2.4).

Proposition 2 *The stationary points of the dynamical system (2.4) consist of point in the set*

$$\{\boldsymbol{F}=\mathbf{0}\}\equiv\{\boldsymbol{w}\ :\ \nabla J^e(\boldsymbol{w})=\mathbf{0}\ \text{or}\ \boldsymbol{w}=\mathbf{0}\},$$

where $\boldsymbol{F}$ is defined in (2.6). Furthermore, if $L=2$, $\boldsymbol{F}(\boldsymbol{w})$ is not differentiable at $\mathbf{0}$; if $L>2$, the Jacobian matrix $\nabla\boldsymbol{F}(\mathbf{0})=0$.

Proposition 3 *Assume that $\langle xx^T\rangle_N$ and $\langle A\rangle_N$ are invertible.*

1. *If $\sigma=0$,*

$$\{\boldsymbol{w}\ :\ \nabla J^e(\boldsymbol{w})=0\}=\left\{(w_x^*,w_y)\ :\ w_y\in\mathbb{R}^{d_y}\right\},\tag{2.10}$$

where $w_x^*=\langle xx^T\rangle_N^{-1}\langle gx\rangle_N$.

2. *If $\sigma\neq 0$,*

$$\boldsymbol{w}^*=\begin{pmatrix}w_x^*\\ w_y^*\end{pmatrix}=\begin{pmatrix}\alpha^*\\ \sigma^{-1}\beta^*\end{pmatrix}=\begin{pmatrix}I_{d_x\times d_x}&0\\ 0&\sigma^{-1}I_{d_y}\end{pmatrix}\langle A\rangle_N^{-1}\begin{pmatrix}\langle gx\rangle_N\\ \langle gy\rangle_N\end{pmatrix}\tag{2.11}$$

is the unique critical point for $\nabla J^e(\boldsymbol{w})$. Furthermore, we have $w_x^*=\alpha^*$ and

$$\begin{pmatrix}\alpha^*\\ \beta^*\end{pmatrix}=\langle A\rangle_N^{-1}\begin{pmatrix}\langle gx\rangle_N\\ \langle gy\rangle_N\end{pmatrix}\tag{2.12}$$

which is independent from σ in data.

We remark that the assumption made in (1.3) implies that $\langle xx^T\rangle_N$ is invertible.

Assumption 1 In $M_\sigma=\begin{pmatrix}X\\ \sigma Y\end{pmatrix}$, X and Y are two independent random vectors where $Y\equiv N(0,I_{d_y})$ and X is a random vector in $\mathbb{R}^{d_x}$ such that $\mathbb{E}[XX^T]$ is invertible.

Analogous to the two-dimensional linear regression problem, the following theorem relates $\|w_y^*\|$ to the standard deviation of the noise and the cardinality of the data set.

Theorem 1 *Suppose that $\sigma\neq 0$ and (x_i,y_i), $i=1:N$ are independently sampled from the distributions X and Y satisfying Assumption 1. Let (w_x^*,w_y^*) denote a stationary point of (2.4). For sufficiently large N, with a high probability,*

$$\|w_y^*\|\le\frac{C_{g,X,Y}}{\sigma\sqrt{N}},$$

and for some $C_{g,X,Y}\ge 0$ which depends only on $g(x)$ and the distribution (X,Y).

Proof Let us denote

$$\langle xx^T\rangle_N=\widetilde{\Sigma}_X\quad\text{and}\quad\langle yy^T\rangle_N=\widetilde{\Sigma}_Y,$$

which are the maximum likelihood estimations of the covariance matrices Σ_X and $\Sigma_Y=I_{d_y}$. Given Σ_X is invertible and N is large enough, we have $\langle A\rangle_N$ and S which are all invertible by matrix perturbation theory [52]. Moreover, we have

$$w_y^*=\sigma^{-1}S^{-1}\left(\langle gy\rangle_N-\langle yx^T\rangle_N\widetilde{\Sigma}_X^{-1}\langle gx\rangle_N\right),$$

by representing A^{-1} in (2.11) in terms of block matrix where

$$S=\widetilde{\Sigma}_Y-\langle yx^T\rangle_N\widetilde{\Sigma}_X^{-1}\langle xy^T\rangle_N.$$

According to the independence of X and Y and the law of large numbers, we have

$$\left[\langle xy^T\rangle_N\right]_{ij} = \mathcal{O}\left(\frac{1}{\sqrt{N}}\right) \quad \text{and} \quad \left[\langle yx^T\rangle_N\right]_{ji} = \mathcal{O}\left(\frac{1}{\sqrt{N}}\right),$$

and

$$\left[\langle gy\rangle_N\right]_j = \mathcal{O}\left(\frac{1}{\sqrt{N}}\right),$$

for $i = 1 : d_x$ and $j = 1 : d_y$. In addition, similar results for correlated matrix [2,11] show that

$$\widetilde{\Sigma}_X = \Sigma_X + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right) \quad \text{and} \quad \widetilde{\Sigma}_Y = I_{d_y} + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$$

with high probability if N is large. Furthermore, we have

$$\langle A\rangle_N \equiv \begin{pmatrix} \langle xx^T\rangle_N & \langle xy^T\rangle_N \\ \langle yx^T\rangle_N & \langle yy^T\rangle_N \end{pmatrix} = \begin{pmatrix} \Sigma_X & 0 \\ 0 & I_{d_y} \end{pmatrix} + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right),$$

and notice

$$S = \widetilde{\Sigma}_Y - \langle yx^T\rangle_N \widetilde{\Sigma}_X^{-1} \langle xy^T\rangle_N = I_{d_y} + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right) + \mathcal{O}\left(\frac{1}{N}\right).$$

This means

$$\|S^{-1}\| \le C_Y\left(1 - \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)\right)^{-1} \quad \text{and} \quad \|\widetilde{\Sigma}_X^{-1}\| \le C_X\left(\|\Sigma_X\|_{\min} - \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)\right)^{-1},$$

where $\|\Sigma_X\|_{\min}$ denotes the minimal singular value of Σ_X and C_X and C_Y are constants depended only on X and Y. Thus, for some $C_{g,X,Y} \ge 0$, we have

$$\|w_y^*\| \le \sigma^{-1}\|S^{-1}\|\left(\|\langle gy\rangle_N\| + \|\langle yx^T\rangle_N\|\left\|\widetilde{\Sigma}_X^{-1}\right\|\|\langle gx\rangle_N\|\right) \le \frac{C_{g,X,Y}}{\sigma\sqrt{N}}.$$

□

Finally, we have the following estimate for w_y^* when the target function $g(x) = \tilde{g}(x) + \mu^T x$ is a perturbation of a linear function $\mu \in \mathbb{R}^{d_x}$.

Corollary 1 *If $g(x) = \tilde{g}(x) + \mu^T x$ and $|\tilde{g}(x)| \le \delta$ for all $x \in \mathbb{R}^{d_x}$, then*

$$\|w_x^* - \mu\| \le b_{X,Y}\delta \quad \textit{and} \quad \|w_y^*\| \le \frac{\delta\, C_{X,Y}}{\sigma\sqrt{N}},$$

for some constants $b_{X,Y}$ and $C_{X,Y}$ depending only on the distribution X and Y. Furthermore,

$$\left|g(x) - (\boldsymbol{w}^*)^T\boldsymbol{x}\right| \le \delta\left(1 + b_{X,Y}\|x\| + \frac{C_{X,Y}\|y\|}{\sigma\sqrt{N}}\right),$$

for any $\boldsymbol{x} = (x, y) \in \mathbb{R}^d$.

The following numerical results in Fig. 1 verify the estimate of $\|w_y^*\|$ in Theorem 1 and the claim in Corollary 1. Here $d_x = 2$ ($x = (x_1, x_2)$), $d_y = 1$, and we take $g_0(x_1, x_2) = \pi(\sin(\pi x_1) + \sin(\pi x_2))$ in the left figure. For the right figure, we have $g_1(x_1.x_2) = 4(x_1 + x_2) + 0.1(\sin(\pi x_1) + \sin(\pi x_2))$, $g_2(x_1.x_2) = 2(x_1 + x_2) + 0.1(\sin(\pi x_1) + \sin(\pi x_2))$, $g_3(x_1.x_2) = \pi(\sin(\pi x_1) + \sin(\pi x_2))$, and $N = 10^6$. We sample the data as $x_1, x_2 \sim U[-1, 1]$ and $y \sim N(0, 1)$ and then compute (w_x^*, w_y^*) by averaging 10 results using (2.11).

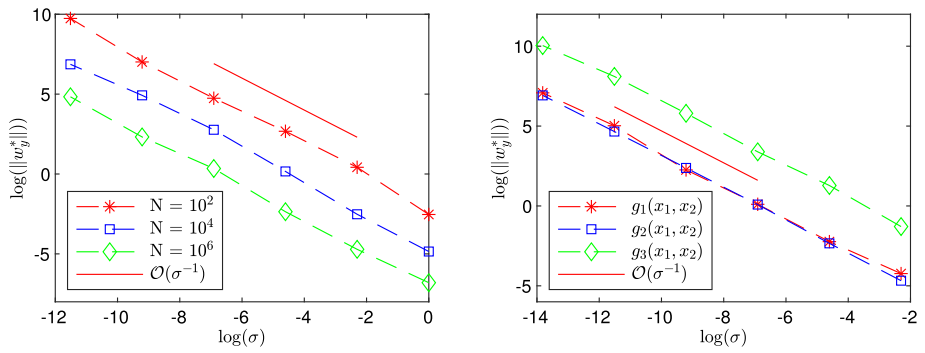

Fig. 1 The log–log diagram of $\|w_y^*\|$ (left) with different σ and N and $\|w_y^*\|$ with different $g(x)$ (right)

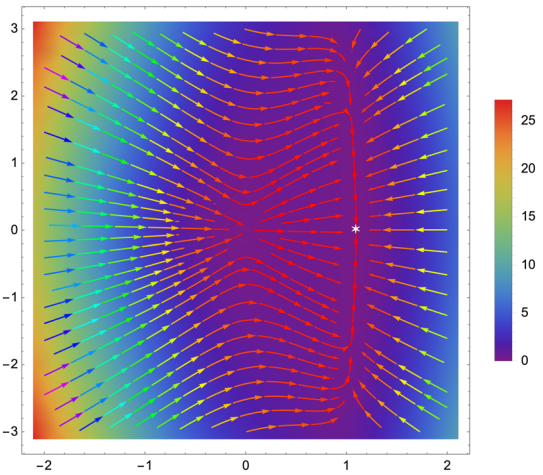

Fig. 2 Stream lines of the system with $\sigma = 0.1$ on the $w_x w_y$-plane. The horizontal and the vertical axes are, respectively, the w_x- and the w_y-axis. We take $L = 5, \sigma = 0.1, N = 10^4$, and $g(x) = x + 0.1\sin(\pi x)$. As shown in Corollary 1, we have $w_x^* \approx 1$ and $w_y^* \approx 6.596 \times 10^{-3} \leq \frac{C_{x,y}}{10\sigma\sqrt{N}}$. The color in the background corresponds to the value of $\|\boldsymbol{F}\|$

2.2 Bifurcation and slow manifold when σ is small

In Proposition (3), we showed that when $\sigma = 0$, the dynamical system (2.4) has a stationary manifold defined as

$$\Gamma_0 := \left\{(w_x^*, w_y) \ : \ w_y \in \mathbb{R}^{d_y}\right\}. \tag{2.13}$$

For small positive σ, Γ_σ degenerates into a single point (w_x^*, w_y^*) denoted as the slow manifold Γ_σ. In this section, we present a phase plane analysis of (2.4) and relate the consequence in training a deep LNN.

In Fig. 2, we present the phase portrait of the dynamical system (2.4) on the $w_x w_y$-plane. We see that $w_x(t)$ first converges to a neighborhood of Γ_σ. Once in the neighborhood, $w_y(t)$ converges to w_y^* on a slower time scale. Asymptotically, $(w_x(t), w_y(t))$ converges to the stationary point (w_x^*, w_y^*). Indeed, the following theorem confirms that Γ_0 and Γ_σ are stable.

Theorem 2 *Suppose that $x_i, y_i, i = 1, 2, \cdots, N$ are independently sampled from distributions X and Y satisfying Assumption 1. Consider the vector field $\boldsymbol{F}$ defined in (2.6).*

- *If $\sigma = 0$, then the eigenvalues of $\nabla \boldsymbol{F}(\boldsymbol{w}^*)$ are non-positive and the associated eigenvectors to the zero eigenvalues are $\{(0, w_y)|w_y \in \mathbb{R}^{d_y}\} = \Gamma_0 - (w_x^*, 0)$ for any $\boldsymbol{w}^* \in \Gamma_0$.*
- *If $\sigma > 0$, $\frac{1}{\sqrt{N}} \ll \sigma$, and $\boldsymbol{w}^*$ is the unique nonzero stationary point, then there are d_y negative eigenvalues of $\nabla \boldsymbol{F}(\boldsymbol{w}^*)$ with scale $\mathcal{O}(\sigma^2)$ with high probability.*

Proof If $\sigma = 0$ and $\boldsymbol{w}^* \in \Gamma_0$, first we have the eigenvalues of $\nabla \boldsymbol{F}(\boldsymbol{w}^*)$ are non-positive as shown in Proposition 3. Moreover, we have

$$\nabla \boldsymbol{F}(\boldsymbol{w}^*) = -\|\boldsymbol{w}^*\|^{-\frac{2}{L}} \boldsymbol{M}(\boldsymbol{w}^*) \nabla^2 J^e(\boldsymbol{w}^*),$$

where

$$\boldsymbol{M}(\boldsymbol{w}^*) = \|\boldsymbol{w}^*\|^2 I + (L-1)\boldsymbol{w}^* (\boldsymbol{w}^*)^T.$$

Recall $\nabla^2 J^e(\boldsymbol{w}^*) = \langle A_0 \rangle_N = \begin{pmatrix} \langle xx^T \rangle_N & 0 \\ 0 & 0 \end{pmatrix}$ and $(\boldsymbol{w}^*)^T \langle A_0 \rangle_N = \langle g\boldsymbol{x} \rangle_N^T = \begin{pmatrix} \langle gx \rangle_N \\ 0 \end{pmatrix}$, thus it follows that

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{w}^*) \nabla^2 J^e(\boldsymbol{w}^*) &= \|\boldsymbol{w}^*\|^2 \langle A_0 \rangle_N + (L-1)\boldsymbol{w}^* \langle g\boldsymbol{x} \rangle_N^T \\ &= \begin{pmatrix} \|\boldsymbol{w}^*\|^2 \langle xx^T \rangle_N + (L-1) w_x^* \langle gx \rangle_N^T & 0 \\ (L-1) w_y^* \langle gx \rangle_N^T & 0 \end{pmatrix}. \end{aligned}$$

Thus, the eigenvectors of $\nabla \boldsymbol{F}(\boldsymbol{w}^*)$ corresponding to zero eigenvalues belong to $\Gamma_0 - (w_x^*, 0)$ since $\nabla \boldsymbol{F}(\boldsymbol{w}^*)$ has the form $\begin{pmatrix} * & 0 \\ * & 0 \end{pmatrix}$.

If $\sigma > 0$ and $\boldsymbol{w}^* \in \Gamma_\sigma$, we still have

$$\nabla \boldsymbol{F}(\boldsymbol{w}^*) = -\|\boldsymbol{w}^*\|^{-\frac{2}{L}} \boldsymbol{M}(\boldsymbol{w}^*) \nabla^2 J^e(\boldsymbol{w}^*)$$

and

$$\boldsymbol{M}(\boldsymbol{w}^*) \nabla^2 J^e(\boldsymbol{w}^*) = \|\boldsymbol{w}^*\|^2 \langle A_\sigma \rangle_N + (L-1)\boldsymbol{w}^* \langle g\boldsymbol{x} \rangle_N^T.$$

In addition, we have

$$\langle A_\sigma \rangle_N = \begin{pmatrix} \langle xx^T \rangle_N & \sigma \langle xy^T \rangle_N \\ \sigma \langle yx^T \rangle_N & \sigma^2 \langle yy^T \rangle_N \end{pmatrix} = \begin{pmatrix} \Sigma_X & 0 \\ 0 & \sigma^2 I_{d_y} \end{pmatrix} + \mathcal{O}\left(\frac{\sigma}{\sqrt{N}}\right).$$

Furthermore, we notice

$$\boldsymbol{w}^* \langle g\boldsymbol{x} \rangle_N^T = \begin{pmatrix} w_x^* \langle gx \rangle_N^T & \sigma w_x^* \langle gy \rangle_N^T \\ w_y^* \langle gx \rangle_N^T & \sigma w_y^* \langle gy \rangle_N^T \end{pmatrix} = \begin{pmatrix} w_x^* \langle gx \rangle_N^T & 0 \\ w_y^* \langle gx \rangle_N^T & 0 \end{pmatrix} + \mathcal{O}\left(\frac{\sigma}{\sqrt{N}}\right).$$

It follows that

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{w}^*) \nabla^2 J^e(\boldsymbol{w}^*) &= \begin{pmatrix} \|\boldsymbol{w}^*\|^2 \Sigma_X + (L-1) w_x^* \langle gx \rangle_N^T & 0 \\ (L-1) w_y^* \langle gx \rangle_N^T & \|\boldsymbol{w}^*\|^2 \sigma^2 I_{d_y} \end{pmatrix} + \mathcal{O}\left(\frac{\sigma}{\sqrt{N}}\right) \\ &=: \boldsymbol{K} + \mathcal{O}\left(\frac{\sigma}{\sqrt{N}}\right). \end{aligned}$$

Here, we notice that there are d_y eigenvalues of $\boldsymbol{K}$ equals $\|\boldsymbol{w}^*\|^2 \sigma^2$ with eigenspace $\Gamma_0 - (w_x^*, 0)$. Given the matrix perturbation theory [52], there exist at least d_y negative eigenvalues of $\nabla \boldsymbol{F}(\boldsymbol{w}^*)$ with scale $\mathcal{O}(\sigma^2)$ if $\frac{\sigma}{\sqrt{N}} \ll \sigma^2$. □

In the regime $0 < \sigma \ll 1$ and $N \gg \sigma^{-2}$, the gradient descent flow (2.4) tend to converge slowly to the optimal parameter $\boldsymbol{w}^*$ due to the gap in the eigenvalues of $\nabla \boldsymbol{F}(\boldsymbol{w}^*)$, as

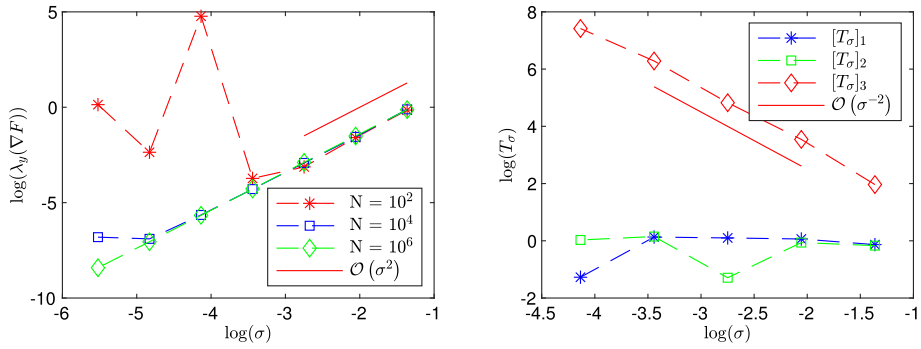

Fig. 3 The log–log diagram of $\lambda_y(\nabla \boldsymbol{F})$ (left) and $[T_\sigma]_i$ for $i = 1:3$ (right), where $g(x_1, x_2) = \pi * (\sin(\pi x_1) + \sin(\pi x_2))$. We observe that indeed the smallest eigenvalue of $\nabla \boldsymbol{F}$ follows the scale of σ^2 when $\frac{1}{\sqrt{N}} \ll \sigma$, while $[T_\sigma]_3$ follows the scale of $\mathcal{O}(\sigma^{-2})$, which confirms that the convergence of the slowest component takes place in the $\mathcal{O}(\sigma^{-2})$ time scale

Theorem 2 shows. We refer to this slow convergence as one of the side effects of learning from embedded data because it stems from the fact that data distribution essentially concentrates on a lower-dimensional manifold.

In Fig. 3, we present a set of numerical simulations demonstrating this slow convergence when N (the number of data points) is sufficiently large. In the experiment, $d_x = 2$ and $d_y = 1$, so Γ_σ is a point on the line $\{(w_x^*, w_y) : w_y \in \mathbb{R}\}$. In the left subplot, we report the smallest eigenvalue of $\nabla \boldsymbol{F}$, corresponding to the direction parallel to Γ_σ, for different N and σ. In the right subplot, we report the quantities

$$[T_\sigma]_i := \inf_t \left\{ t \ : \ \left\| [\boldsymbol{w}_\sigma]_i(t) - [\boldsymbol{w}_\sigma^*]_i \right\| \le 10^{-6} \right\},$$

where $[\boldsymbol{w}_\sigma]_i(t)$ stands for the ith component of $\boldsymbol{w}_\sigma$ at the time t and $\boldsymbol{w}_\sigma^*$ is the nonzero stationary point as in Proposition 3. $[T_\sigma]_i$ gives the first time that the ith component of $\boldsymbol{w}_\sigma$ becomes within 10^{-6} distance to $[\boldsymbol{w}_\sigma^*]_i$. We now focus on the convergence of the third component, corresponding to w_y. Assuming that $\boldsymbol{w}_\sigma(0)$ is in a sufficiently close neighborhood of $\boldsymbol{w}_\sigma^*$ so that linear theory applies. We then have $\|[\boldsymbol{w}_\sigma]_3(t) - [\boldsymbol{w}_\sigma^*]_3\| \le e^{-C\lambda_y(\nabla \boldsymbol{F})t}$ which means $[T_\sigma]_3 \sim C\sigma^{-2}$. This indicates that the time to reach within a small distance of $\boldsymbol{w}^*$ is proportional to $1/\sigma^2$. Numerical results in Fig. 3 verify the slow convergence phenomenon. Here, $\boldsymbol{w}_\sigma$ is computed by simulating the system (2.4) directly with *ode45* in MATLAB with time step size 5×10^{-3}. Correspondingly, it takes $200 \times e^{[T_\sigma]_i}$ iterations in *ode45* such that the ith component of $\boldsymbol{w}_\sigma$ becomes within 10^{-6} distance to $[\boldsymbol{w}_\sigma^*]_i$.

The following proposition shows that a similar gap in the eigenvalues may exit even for systems defined with relatively small number of data points.

Proposition 4 *Under that same conditions in Theorem 2 with* $0 < \sigma \ll 1$, *for any* $N \ge 1$ *and* $\left\| \|\boldsymbol{w}^*\|^2 \langle xy^T \rangle_N + (L-1) w_x^* \langle gy \rangle_N^T \right\| \le C$, *denoting* $\lambda(\cdot)$ *as the spectrum of a matrix and* $\nabla \boldsymbol{F}(\boldsymbol{w}^*) = -\|\boldsymbol{w}^*\|^{-\frac{2}{L}} \begin{pmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{pmatrix}$, *where*

$$F_{11} = \|\boldsymbol{w}^*\|^2 \langle xx^T \rangle_N + (L-1) w_x^* \langle gx \rangle_N^T, \qquad F_{12} = \sigma \left(\|\boldsymbol{w}^*\|^2 \langle xy^T \rangle_N + (L-1) w_x^* \langle gy \rangle_N^T \right),$$
$$F_{21} = (L-1) w_y^* \langle gx \rangle_N^T + \sigma \|\boldsymbol{w}^*\|^2 \langle yx^T \rangle_N, \qquad F_{22} = \|\boldsymbol{w}^*\|^2 \sigma^2 \langle yy^T \rangle_N + \sigma w_y^* (L-1) \langle gy \rangle_N^T,$$

then $\lambda(\nabla \boldsymbol{F}(\boldsymbol{w}^*)) \subset G_1 \cup G_2$, *where*

$$G_i = \lambda(F_{ii}) \cup \left\{ \lambda \notin \lambda(F_{ii}) \,\middle|\, \left\| (F_{ii} - \lambda I)^{-1} \right\|^{-1} \le \|F_{ji}\| \right\}, i = 1:2, j \neq i.$$

More precisely, for $i = 2$, *we have*

$$G_2 = \lambda(F_{22}) \cup \left\{ \lambda \notin \lambda(F_{22}) \,\middle|\, \left\| (F_{22} - \lambda I)^{-1} \right\|^{-1} \le \|F_{12}\| \right\}.$$

In particular, since $\|F_{12}\| \le \sigma \left\| \|\boldsymbol{w}^*\|^2 \langle xy^T \rangle_N + (L-1) w_x^* \langle gy \rangle_N^T \right\| \le \sigma C$ *and* $\lambda(F_{22}) \sim \mathcal{O}(\sigma)$, *it follows that* $\lambda \sim \mathcal{O}(\sigma)$ *for any* $\lambda \in G_2$.

The proof of this theorem is a quick application of Gershgorin's theorem for block matrices [54]. Following this proposition, $\nabla \boldsymbol{F}(\boldsymbol{w}^*)$ may have eigenvalues falling in the set G_2. In that case, the magnitudes of those eigenvalues are $\mathcal{O}(\sigma)$. Hence, the proposition can be applied to understand the flow in a mini-batch stochastic gradient descent algorithm. Each step of SGD can be understood as one discrete step of (2.4) with a relatively small N corresponding to the mini-batch size. Thus, this proposition suggests that employing SGD in training can be more efficient, as the eigenvalues of the smallest amplitude scale as $\mathcal{O}(\sigma)$ instead of $\mathcal{O}(\sigma^2)$ (if $N \gg \sigma^{-2}$), although it will not always avoid the slow convergence caused by the small variance σ in the y-directions. See Fig. 5 for a supporting numerical study.

2.3 Slow convergence

In this subsection, we show that deep LNNs may have yet another hindrance to convergence, depending on the initialization. The following theorem shows that the trajectories of (2.4) may be attracted to a neighborhood of the origin, and if that happens, it will take a very long time to escape.

Theorem 3 *Assume* $0 < C_1 \le \langle A_\sigma \rangle_N \le C_2$ *and* $\|\langle g\boldsymbol{x} \rangle_N\| = \mathcal{O}(1)$, *then for* $\epsilon \ll 1$ *we have*

$$T_L(\epsilon) := \inf \left\{ t \,:\, \|\boldsymbol{w}(0)\| = \epsilon,\ \|\boldsymbol{w}(t) - \boldsymbol{w}(0)\| \ge \frac{\epsilon}{2} \right\} \ge C \epsilon^{\frac{2}{L} - 1}, \tag{2.14}$$

where $\boldsymbol{w}(t)$ *is solution of* (2.4) *and* C *depends on* L, $\langle A_\sigma \rangle_N$, *and* $\langle g\boldsymbol{x} \rangle_N$.

For brevity, this theorem shows that deeper LNNs require more time for convergence if the initialization is very close to the origin or the training process reaches the near field of the origin. In practice, a commonly accepted heuristics is to avoid initializing weights near the origin. The above theorem provides a theoretical interpretation for that heuristics, at least in the context of training deep linear networks. However, as shown in Figs. 2 and 5, even if one initializes the weights to be far from the origin, the weights can be attracted to a neighborhood of the origin during the gradient flow. This phenomenon, which has not been discovered before, can still cause the slow convergence in training LNNs.

Figure 4 demonstrates the convergence issues corresponding to Theorem 2 and Theorem 3. Here, we simulate the dynamical system (2.4), with $L = 10$ and $\boldsymbol{w}(0) = (-2, 1)$. The data are sampled as follows: $\boldsymbol{x}_i = (x_i, \sigma y_i) \in \mathbb{R}^2$, $x_i \sim U[-1, 1]$, $y_i \sim N(0, 1)$, $g(x) = \pi \sin(\pi x)$, $N = 10^4$, and $\sigma = 0.05$.

Furthermore, in Fig. 5, we also observe similar results when we train a LNN with the full gradient descent method with a special initialization that $[W^\ell]_{i,j}$ is a fixed constant for each i, j such that $\|\boldsymbol{w}\| = 2^{-6}$. This initialization can satisfy the condition in (2.5) as required in [3] to make the dynamic system (2.4) as the continuous limit of the FGD method. Thus,

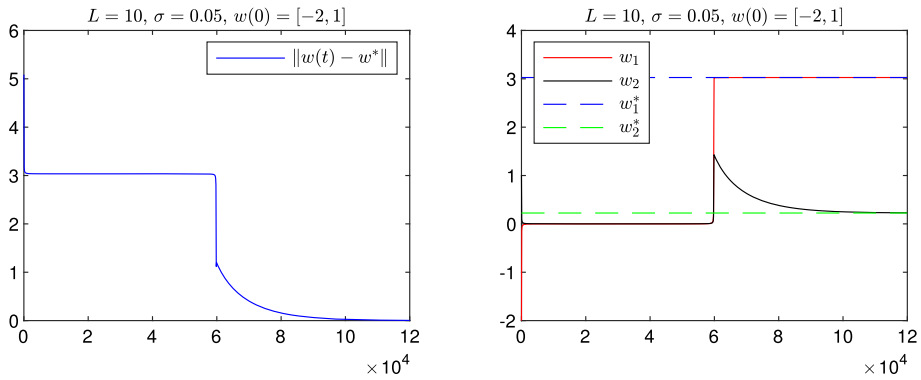

Fig. 4 Convergence of $\boldsymbol{w}(t) = (w_x(t), w_y(t))$. We see that the trajectory is attracted to the origin and stay a long time ($\sim 6 \times 10^4$ iterations) before escaping. Furthermore, once flowing pass the origin, $w_x(t)$ quickly converge to a small neighborhood of the slow manifold Γ_σ. But it will take another long period before $\boldsymbol{w}(t)$ gets close to the optimal point due to the slow convergence of $w_y(t)$

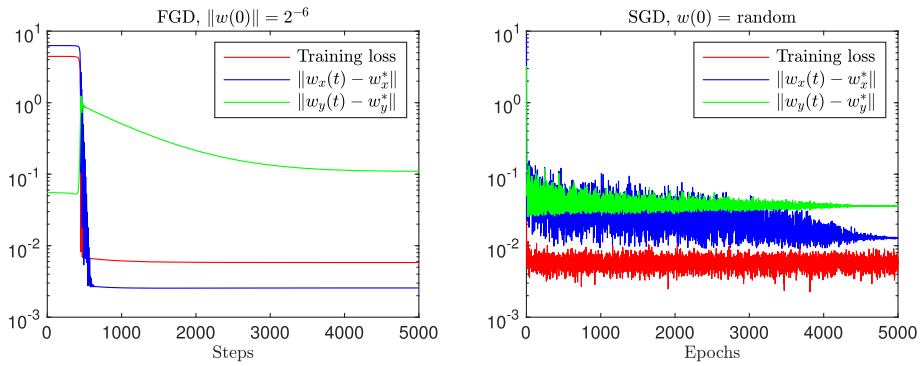

Fig. 5 The convergence of the loss function, $\|w_x(t) - w_x^*\|$, and $\|w_y(t) - w_y^*\|$ when $\sigma = 0.05$. The FGD results match the previous analysis very well. Given the initialization that $\|\boldsymbol{w}\| = 2^{-6}$, it gets stuck around the origin for a while and then w_x converges very quick while w_y converges very slowly after escaping the origin. In the SGD results, a suitable random initialization strategy is important to the success of the SGD in DNNs [13,23,28]. Generally, it is hard to notice the trapping issue around the origin for SGD with random initialization. However, as shown here and in Proposition 4, w_y converges slowly for both FGD and SGD

we take a full gradient descent training algorithm with a decreasing learning rate from 2.5×10^{-3} to 2.5×10^{-5} under a cosine annealing schedule [37]. In addition, we take $L = 6$ and $n = 10$ for this LNN. The training data are created by taking $d_x = 3$ and $d_y = 2$, $\boldsymbol{x}_i = (x_i, \sigma y_i)$, $x_i \sim U\left([-1, 1]^{d_x}\right)$, $y_i \sim N(0, I_{d_y})$, and $g(x) = 2\sum_{i=1}^{3}[x]_i + 0.1\sum_{i=1}^{3}\sin(\pi[x]_i)$, $N = 4 \times 10^3$, and $\sigma = 0.05$. Moreover, we are also interested in how SGD will perform under this situation. We apply SGD for the same LNN and training data with Kaiming's initialization [28] for W^ℓ and a mini-batch size 50. We also show the results in Fig. 5.

Related work. Theorem 2 shows that (2.4) has a slow manifold Γ_σ and the convergence of $w_y(t)$ to w_y^* takes place in the $\mathcal{O}(\sigma^{-2})$ time scale. Similar results about the slow convergence (in the components corresponding to small singular values in the data matrix) are also reported in [51] for randomized Kaczmarz iterations and [25] for gradient descent in neural networks. In the setting of this paper, if $\sigma\sqrt{N} \ll 1$ and $\tilde{g}$ is not small enough, then Corollary 1 shows that $\|w_y^*\| \gg 1$. In this case, "early stopping" [56] may be employed to

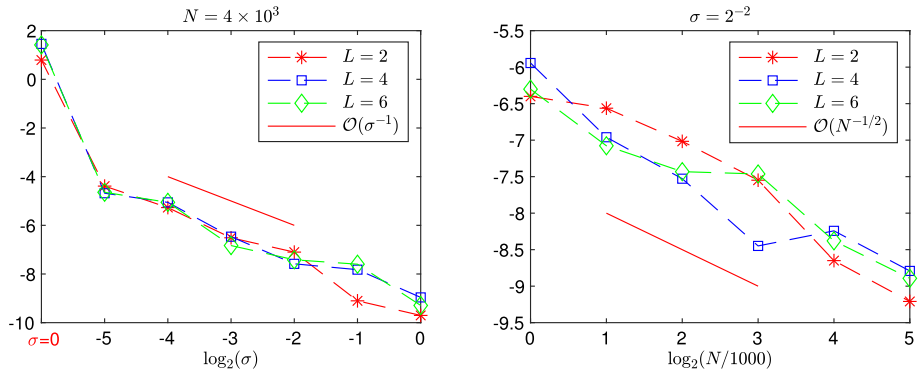

Fig. 6 $\log_2(\|w_y\|)$ of trained LNNs for $L = 2, 4, 6$. Here we still observe that $\|w_y^*\| = \mathcal{O}\left(\sigma^{-1}N^{-1/2}\right)$

control $\|w_y(T)\|$. The similar results can also be found in [38], which presents that small eigenvalues for the associated Gram matrix make the convergence of gradient descent very slow. In that case, the slow convergence gives us ample time to stop the training process and obtain solutions with good generalization property. On the other hand, Corollary 1 and Theorem 2 also indicate that there exist some cases in which the early stopping is not recommended. For example, $\|w_y^*\|$ could be small if $\sigma\sqrt{N} \gg 1$ and $\tilde{g}$ in Corollary 1 is relatively small.

2.4 Regularization effects of noise and network's depth

2.4.1 Regularization effect of noise

Theorem 1 states that the presence of noise in the y-components, i.e., $\sigma \neq 0$, can result in w_y^* with a small amplitude, provided that the training data set is sufficiently large. Moreover, if the noise scale is fixed in data $\boldsymbol{x}_i$, Theorem 1 presents that more data are needed to control the amplitude of $\|w_y^*\|$. Figure 6 demonstrates these results in training LNN models using SGD.

In Fig. 6, we notice that $w_y(t)$ is non-constant even when $\sigma = 0$. It is due to the "mixing" that comes from the multiple hidden layers and can be seen from (2.4) (more explicitly from (2.15)). This is different from pure linear regression case where w_y will keep constant after initialization. Given this observation, we will further study the properties of training LNNs when $\sigma = 0$ in the next subsection.

The basic setup is same to what we have done in Fig. 5. Noticing that w_y in LNNs may be difficult to converge when σ is small, we test only $\sigma = 2^k$ for $k = 0 : -5$. Thus, we apply SGD only 500 epochs for these experiments and the reported values of $\|w_y\|$ are obtained by averaging over 5 individual tests.

2.4.2 The regularization and side effects of depth when $\sigma = 0$

In this subsection, we focus on the setting where the training data lie on the low-dimensional manifold $\mathcal{M}$ exactly, i.e., $\boldsymbol{x}_i \sim M_0$. We prove that the size of $\|w_y^*\|$ trained with this data may decrease as the depth of the network increases, for the initial value $\boldsymbol{w}(0)$ in certain subregion of $\mathbb{R}^d$.

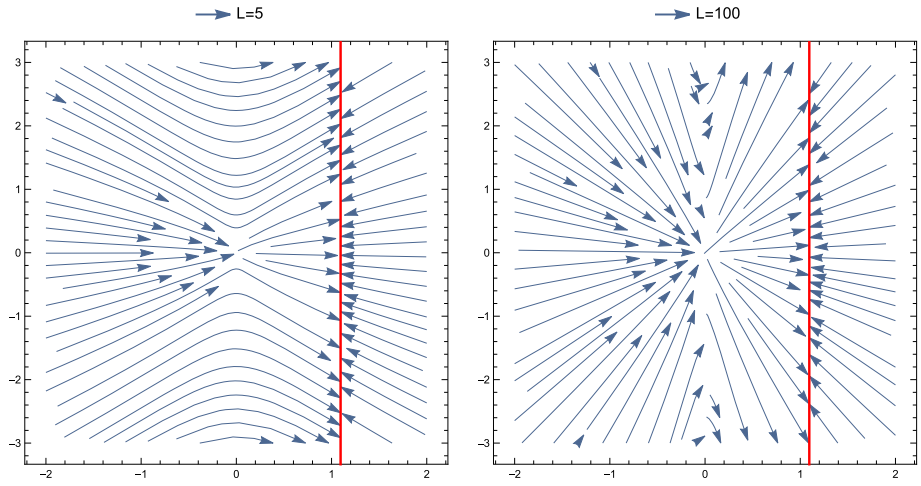

Fig. 7 Phase portraits of the system (2.15) on the $w_x w_y$-plane with $L = 5$ (left) and $L = 100$ (right)

Since $\boldsymbol{x}_i \sim M_0$, we have the data points $\boldsymbol{x}_i = (x_i, 0) \in \mathbb{R}^{d_x+d_y}$ and $g_i \in \mathbb{R}$. Under this situation, the loss function will degenerate to

$$J^e(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=1}^{N}(w_x^T x_i - g_i)^2,$$

where

$$\frac{\partial J^e(\boldsymbol{w})}{\partial w_x} = \langle xx^T\rangle_N w_x - \langle gx\rangle_N \quad \text{and} \quad \frac{\partial J^e(\boldsymbol{w})}{\partial w_y} = 0.$$

Equations of $\boldsymbol{w}$ in (2.4) are reduced to

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}t}w_x &= f(w_x, w_y) = -\|\boldsymbol{w}\|^{-\frac{2}{L}}\left(\|\boldsymbol{w}\|^2\frac{\partial J^e(\boldsymbol{w})}{\partial w_x} + (L-1)\left(w_x^T\frac{\partial J^e(\boldsymbol{w})}{\partial w_x}\right)w_x\right), \\ \frac{\mathrm{d}}{\mathrm{d}t}w_y &= g(w_x, w_y) = -(L-1)\|\boldsymbol{w}\|^{-\frac{2}{L}}\left(\left(w_x^T\frac{\partial J^e(\boldsymbol{w})}{\partial w_x}\right)w_y\right), \end{cases} \tag{2.15}$$

since $\boldsymbol{w}^T\frac{\partial J^e(\boldsymbol{w})}{\partial \boldsymbol{w}} = w_x^T\frac{\partial J^e(\boldsymbol{w})}{\partial w_x}$.

According to Proposition 2, the stationary points of the above system consist of $\mathbf{0}$ and

$$\Gamma_0 = \left\{(w_x^*, w_y) \ : \ w_y \in \mathbb{R}^{d_y}\right\}$$

where we assume w_x^* is the unique solution of $\frac{\partial J^e(\boldsymbol{w})}{\partial w_x} = \langle xx^T\rangle_N w_x - \langle gx\rangle_N = 0$.

In the following, we study the relationship between L, the network's depth, and $\frac{\partial f}{\partial y} = w_y^*$. Naturally, the smaller the magnitude of $\frac{\partial f}{\partial y}$, the more consistent the network's output would be when the testing data deviates from the training data manifold.

To begin this study, we first show the following diagram about the phase portraits of the system (2.15) on the $w_x w_y$-plane with $L = 5$ and $L = 100$.

According to the above phase portraits, if $\boldsymbol{w}$ is initialized on the right of Γ_0 (the red line in Fig. 7), we have $|w_y^*| \le |w_y(0)|$, which can be understood as the regularization effect of the LNN structure since $w_y^* = w_y(0)$ in classical linear regression model when $\sigma = 0$. In addition, we also notice that $|w_y^*| \ge |w_y(0)|$ if $\boldsymbol{w}$ is initialized between the y-axis and Γ_0. This aspect of training can be interpreted as a side effect of the LNN structure comparing to the linear regression case. We now present generalization of this regularization and side effects.

Again, let $\boldsymbol{w} = (w_x, w_y) \in \mathbb{R}^{d_x+d_y}$ with $d_x, d_y \geq 1$. First, we define

$$E_x := \left\{ w_x \in \mathbb{R}^{d_x} \ : \ w_x^T \frac{\partial J^e(\boldsymbol{w})}{\partial w_x} = 0 \right\} \subset \mathbb{R}^{d_x}. \tag{2.16}$$

E_x is an ellipsoid of dimension $d_x - 1$ centered at $\langle xx^T \rangle_N^{-1} \langle gx \rangle_N / 2 = w_x^*/2$, since

$$w_x^T \frac{\partial J^e(w)}{\partial w_x} = w_x^T \langle xx^T \rangle_N w_x - w_x^T \langle gx \rangle_N$$

and $\langle xx^T \rangle_N$ is a symmetric positive definite matrix.

We denote the cylinder generated by E_x as

$$E := E_x \times \mathbb{R}^{d_y} \tag{2.17}$$

and the enclosed region as

$$E^- := \left\{ (w_x, w_y) \ : \ w_x^T \frac{\partial J^e(\boldsymbol{w})}{\partial w_x} < 0, \quad w_y \in \mathbb{R}^{d_y} \right\}.$$

E^- can be regarded as the generalization of the region between y-axis and Γ_0 as in Fig. 7, a region in which $||w_y(t)||$ increases following the flow of (2.15).

To define an analogy to global flow structure of (2.15) depicted in Fig. 7, we introduce the hyperplane

$$H \equiv \left\{ \boldsymbol{w} \in \mathbb{R}^d \ : \ \left(n_E^*\right)^T \left(\boldsymbol{w} - (w_x^*, 0)\right) = 0 \right\},$$

where n_E^* denotes the exterior normal direction of E at $(w_x^*, 0)$ in $\mathbb{R}^d$. Thus, H is the tangent plane of E at $(w_x^*, 0)$ in $\mathbb{R}^d$, separating R^d into two disjoint open sets (half spaces). We denote U^- as the part which contains $(0, 0)$ while U^+ as the other part. More precisely,

$$U^- := \{ \boldsymbol{w} \ : \ {n_E^*}^T \left(\boldsymbol{w} - (w_x^*, 0)\right) < 0 \},$$
$$U^+ := \{ \boldsymbol{w} \ : \ {n_E^*}^T \left(\boldsymbol{w} - (w_x^*, 0)\right) > 0 \}.$$

Here, we also notice that

$$\mathbb{R}^d = U^- \cup H \cup U^+, \ \ (0, 0) \in E \subset \overline{U^-}, \quad \Gamma_0 \subset H.$$

We remark that $\Gamma_0 = H \iff d_x = 1$. Figure 8 illustrates a corresponding diagram for the case $d_x = 2$ and $d_y = 1$.

Assumption 2 Let $\boldsymbol{w}(t)$ be a solution of (2.15) with $\boldsymbol{w}(0)$, and $\boldsymbol{w}(t) \cap E = \emptyset$ for any $0 \leq t \leq T$.

The following proposition states that Assumption 2 holds for some positive time under some conditions on the location of $\boldsymbol{w}(0)$ and the data.

Proposition 5 *If $\boldsymbol{w}(0) \in E^-$ and the correlation matrix of X, Σ_X, satisfies $\Sigma_X = cI_{d_x}$ for some positive constant $c > 0$, then $\boldsymbol{w}(t) \in \overline{E^-}$ for all $0 \leq t \leq T_X$, where $T_X := \inf\{t : \|w_x(t) - w_x^*\| \leq \frac{2\sqrt{3}}{c} \|w_x^*\| \|\Sigma_X - \langle xx^T \rangle\|\}$.*

Since $\|\Sigma_X - \langle xx^T \rangle\| = \mathcal{O}(\frac{1}{N})$ can be made arbitrary small if one increases the number of data points N, in that case, $\boldsymbol{w}(t)$ will stay in $\overline{E^-}$ before it reaches a neighborhood of the stationary manifold Γ_0 (when $\boldsymbol{w}(0) \in E^-$).

Lemma 1 *Suppose that $w_y(0) \neq 0$ and $\boldsymbol{w}(t)$ satisfies Assumption 2 for $0 \leq t \leq T$. Then,*

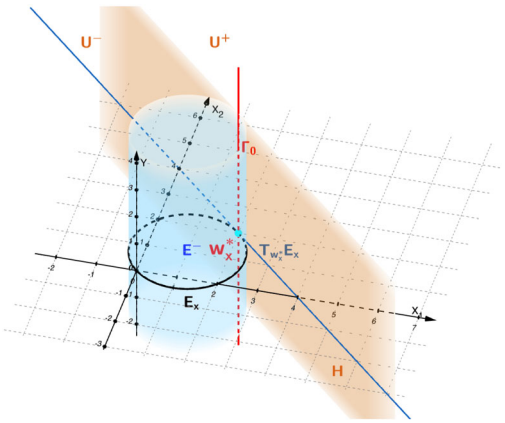

Fig. 8 Example for $d_x = 2$ and $d_y = 1$

1. *if* $\boldsymbol{w}(0) \in U^+$,

$$\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2 \le 0 \ \textit{and} \ \frac{\mathrm{d}}{\mathrm{d}t}\frac{\|w_x(t)\|^2}{\|w_y(t)\|^2} \le 0,$$

2. *if* $\boldsymbol{w}(0) \in E^-$,

$$\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2 \ge 0 \ \textit{and} \ \frac{\mathrm{d}}{\mathrm{d}t}\frac{\|w_x(t)\|^2}{\|w_y(t)\|^2} \ge 0,$$

for $0 \le t \le T$.

As a consequence of the monotonicity of $\|w_x\|^2$ and $\frac{\|w_x\|^2}{\|w_y\|^2}$, we have the next main theorem about the regularization and side effects of LNNs.

Theorem 4 *Suppose that* $w_y(0) \neq 0$ *and* $\boldsymbol{w}(t)$ *satisfies Assumption* 2 *for* $0 \le t \le T$. *Then,*

1. *if* $\boldsymbol{w}(0) \in U^+$,

$$\|w_y(T)\|^2 - \|w_y(0)\|^2 \le \frac{(L-1)\|w_y(0)\|^2}{L\|w_x(0)\|^2 + \|w_y(0)\|^2}\left(\|w_x(T)\|^2 - \|w_x(0)\|^2\right) \le 0; \tag{2.18}$$

2. *if* $\boldsymbol{w}(0) \in E^-$,

$$\|w_y(T)\|^2 - \|w_y(0)\|^2 \ge \frac{(L-1)\|w_y(T)\|^2}{L\|w_x(T)\|^2 + \|w_y(T)\|^2}\left(\|w_x(T)\|^2 - \|w_x(0)\|^2\right) \ge 0. \tag{2.19}$$

Recall that for LNNs, w_y determines the Lipschitz bound of the trained network function, f_θ in the direction orthogonal to the data manifold. Therefore, the first case in Theorem 2.18 can be interpreted as the regularization effect of LNNs: under the stated conditions, the $w_y(T)$ is smaller than $w_y(0)$, implying that the variation of f_θ in the y-directions will reduce. On the other hand, the second case in Theorem 2.18 reveals a "side effect" of LNNs that the variation of f_θ in the y-directions will increase; i.e., the stability (for out of distribution evaluations) of the network will reduce as training progresses.

Furthermore, we can derive the following *a priori* estimate:

$$\|w_y(T)\|^2 \le \|w_y(0)\|^2 + \frac{(L-1)\|w_y(0)\|^2}{L\|w_x(0)\|^2 + \|w_y(0)\|^2}\left(\|w_x^*\|^2 - \|w_x(0)\|^2\right) \tag{2.20}$$

from (2.18). By reorganizing (2.20), we have the following a priori estimate:

$$\left\|w_y(T)\right\|^2 \le h(L)\left(\|w_x(0)\|^2 - \|w_x^*\|^2\right) + \left(\frac{\|w_x^*\|^2}{\|w_x(0)\|^2}\right)\|w_y(0)\|^2, \tag{2.21}$$

where $h(L) = \frac{1+\|w_y(0)\|^2/\|w_x(0)\|^2}{L(\|w_x(0)\|^2/\|w_y(0)\|^2)+1}$ is a decreasing function in terms of L. That is, the upper bound for $\|w_y(T)\|$ with $L = 100$ is smaller than the case of $L = 5$ under the same initialization. Thus, the estimate in (2.21) can partially explains the phenomenon in Fig. 7 in the right of Γ_0 that $|w_y(T)|$ with $L = 100$ is smaller than the case of $L = 5$ under the same initial when $\boldsymbol{w}(t)$ achieves Γ_0.

3 ReLU-activated networks

In this section, we analyze the stability for ReLU deep neural networks (DNNs) when data are sampled from $\mathcal{M}$, i.e., $\boldsymbol{x}_i \sim M_0$. We first show how the low-dimensional data will affect the training process. Given that, we establish the stability estimate for ReLU DNNs with one hidden layer ($L = 2$). By using the recursive structure of ReLU DNNs, we finally prove the stability estimate for deep cases.

As defined in (1.4) and (1.6), we have the ReLU DNN function with $L-1$ hidden layers as

$$\begin{cases} f^\ell(\boldsymbol{x}) &= W^\ell\alpha(f^{\ell-1}(\boldsymbol{x})) + b^\ell, \quad \ell = 2:L, \\ f(\boldsymbol{x},\theta) &= f^L(\boldsymbol{x}), \end{cases} \tag{3.1}$$

where $f^1(\boldsymbol{x}) = W^1\boldsymbol{x} + b^1$, $\alpha = \text{ReLU}$, $W^\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$, $b^\ell, f^\ell \in \mathbb{R}^{n_\ell}$ with $n_0 = d = d_x + d_y$ and $n_L = 1$. Here, W^1 is a $n_1 \times (d_x + d_y)$ matrix, and for the convenience of exposition, we write $W^1 = \begin{pmatrix} W_x^1 & W_y^1 \end{pmatrix}$, where W_x^1 and W_y^1 are, respectively, $n_1 \times d_x$ and $n_1 \times d_y$ matrices. With the data of the form prescribed in Sect. 1.1, we assume $Q = I_d$ and have

$$W^1\boldsymbol{x}_i + b^1 = \begin{pmatrix} W_x^1 & W_y^1 \end{pmatrix}\begin{pmatrix} x_i \\ \sigma y_i \end{pmatrix} + b^1.$$

Then, the loss function is defined as

$$J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(f(\boldsymbol{x}_i;\theta) - g_i)^2, \tag{3.2}$$

where $\boldsymbol{x}_i \sim M_\sigma$ and $\theta = \{W^1, b^1, \cdots, W^L, b^L\}$ denotes all parameters in ReLU DNNs.

If $\boldsymbol{x}_j \in \mathcal{M}$, the key observation here is that

$$\frac{\partial J}{\partial \widetilde{W}_y^1} = 0, \quad \widetilde{W}_y^1 = [W^1 Q]_y.$$

Furthermore, according to the gradient descent update of W^1, we have

$$W^1 Q \leftarrow W^1 Q - \eta\frac{\partial J}{\partial W^1}Q \Longrightarrow W^1 Q \leftarrow W^1 Q - \eta\frac{\partial J}{\partial (W^1 Q)}.$$

Thus, W_y^1 or $\widetilde{W}_y^1$ will not change for any pure gradient descent-based training algorithms. Therefore, without loss of generality, we shall assume in the remaining of this section that $Q = I_d$.The results can be easily extended to $\widetilde{W}^1 = W^1 Q$ and $(\widetilde{x}, \widetilde{y}) = Q^T\boldsymbol{x}$ if $Q \neq I_d$.

Lemma 2 *If $\boldsymbol{x}_j \sim M_0$ in the training data and either the full gradient descent or stochastic gradient descent training algorithm is applied to (3.1) and (3.2), then the following conclusions hold.*

1. *W_y^1 in $W^1 = \left(W_x^1, W_y^1\right)$ will not change during the training process (1.8).*
2. *If there is a ℓ^2 regularization term $\lambda\|\theta\|_{\ell^2}^2$ with an appropriate λ, then W_y^1 will decay to 0.*

Although Lemma 2 also holds for LNNs, estimating $\|\boldsymbol{w}_y^*\|$ directly for LNNs as in Theorem 4 is a more precise and efficient approach to bound the stability metric. However, there is no such linear structure that we can use for ReLU DNNs. Thus, we notice the first consequence in Lemma 2 which shows an invariant property of weights W_y^1 in training ReLU DNNs with $\sigma = 0$ for both full and stochastic gradient descent methods. The invariant property of W_y^1 in training ReLU DNN with (stochastic) gradient descent method plays a critical role in analyzing the stability metric which will be detailed explained in the remaining subsections. For simplicity, we denote $W^\ell(b^\ell)$ and as the initialized weights and $\overline{W}^\ell$ $(\overline{b}^\ell)$ as the weights (biases) after training. In the following, we will use θ^* to denote the parameter set obtained after training. From the discussion above, $\theta^* = \left\{(\overline{W}^\ell, \overline{b}^\ell)\right\}_{\ell=1}^L$ while $\overline{W}^1 = (\overline{W}_x^1, W_y^1)$ due to the conclusion in Lemma 2 if θ^* is obtained by FGD or SGD.

An example. We train and obtain a neural network classifier, $f_{\theta^*} : \mathbb{R}^{784} \mapsto \mathbb{R}^{10}$, using the MNIST data set [19]. The first layer of the network is fully connected. Each image in the MNIST data set is a black-and-white image consisting of 28×28 pixels, and it is regarded as a point in $\mathbb{R}^{784}$. Let $\bar{\boldsymbol{x}}$ be the mean of the data points. Let the unit vector $\boldsymbol{v}_n$ denote a direction corresponding to the least eigenvalue of the covariance matrix. The ratio between the largest and the least eigenvalue of the covariance matrix of MNIST is 5.26×10^{16}. We shall regard the data manifold $\mathcal{M}$ to be the subspace, centered at $\bar{\boldsymbol{x}}$, spanned by the first 783 principal directions.

Let $\overline{W}^1$ denote the weights in f_{θ^*} that connects to the input vector. We introduce perturbation to the weight set $\overline{W}^1 + sW_y$, where $W_y := \boldsymbol{v}_n\boldsymbol{v}_n^T$, and denote the corresponding perturbed network as $f_{\theta_s^*}$. Let $f_{\theta^*}^{[2]}$ denote the second component of the output vector that corresponds to the digit '2'. Classification of an input image $\boldsymbol{x}$ is performed by the maximal component of Softmax$(f_{\theta^*}(\boldsymbol{x}))$, using a trained network f_{θ^*} with 98.14% testing accuracy. In Figure 9, we show the function

$$I_j(s,t) := ||f_{\theta^*}^{[2]}(\boldsymbol{x}_j) - f_{\theta_s^*}^{[2]}(\boldsymbol{x}_j + t\boldsymbol{v}_n)||_2^2, \tag{3.3}$$

for $\boldsymbol{x}_j$. We observe that $I(s,0)$ remains 0 as s varies; in other words, variations in the W_y component of $\overline{W}^1$ does not change the perturbed network's output when evaluated at $\boldsymbol{x}_j$. This means that the data point $\boldsymbol{x}_j$ has no role in the optimization of W_y in W^1, in a gradient descent-based training. Furthermore, $f_{\theta_s^*}$ starts to deviate from f_{θ^*} only when one introduces perturbation to the input $\boldsymbol{x}_j$ in the direction, $\boldsymbol{v}_n$, normal to the data set.

In following subsections, we derive upper bounds on the effect of the perturbation discussed above.

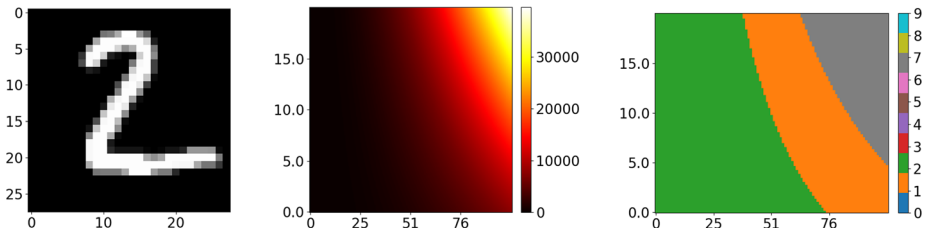

Fig. 9 Left: An image corresponding to the digit '2'. Center: The change in the third component of the output vector, resulting from the perturbation to the input image (parameter t in (3.3), horizontal axis) and to the trained network's first layer weights (parameter s in (3.3), vertical axis). Right: The classification based on the perturbed output.

3.1 Stability estimate for $L = 2$

First, let us consider networks with only one hidden layer, which means $L = 2$. For input training data, we have $\boldsymbol{x}_i = (x_i, 0) \sim M_0$. In addition, we also denote $\Omega_x = (-1, 1)^{d_x}$ as the domain of input of x_i. That is, we have

$$f(\boldsymbol{x}; \theta^*) = f(x, y) := \sum_{i=1}^{n} \overline{W}_i^2 \alpha(\overline{W}_{i,x}^1 x + \overline{b}_i^1 + W_{i,y}^1 y) + \overline{b}^2 \tag{3.4}$$

as the approximation of $g(x)$ after training. According to Lemma 2, $W_{i,y}^1$ is given by initialization since $\sigma = 0$ in the training data.

Then, for any $y \neq 0$, we propose to estimate the following deviation along the y-direction

$$\|f(\cdot, y) - f(\cdot, 0)\|_{L^2(\Omega_x)}^2 = \left\| \sum_{i=1}^{n} e_i(\cdot, y) \right\|_{L^2(\Omega_x)}^2,$$

where

$$e_i(x, y) = \overline{W}_i^2 \left(\alpha(\overline{W}_{i,x}^1 x + \overline{b}_i^1 + W_{i,y}^1 y) - \alpha(\overline{W}_{i,x}^1 x + \overline{b}_i^1) \right). \tag{3.5}$$

In other words, $e_i(x, y)$ describes the stability of each neuron's activation in the first hidden layer.

Using the property of ReLU function, one can easily describe the support of $e_i(x, y)$ given the trained parameters $\overline{W}_{i,x}^1$, $W_{i,y}^1$, and $\overline{b}_i^1$. See the strip depicted in Fig. 13. Thus, we have the following estimate for $e_i(x, y)$.

Lemma 3 *Let $\boldsymbol{x}_i \sim M_0$ in the training data, $f(x, y)$ be a network with a single hidden layer ($L = 2$) defined in (3.4) and trained by FGD or SGD, and $e_i(x, y)$ be defined in (3.5). For any $i = 1 : n_1$, we have*

$$\|e_i(\cdot, y)\|_{L^2(\Omega_x)}^2 \leq \frac{\|\nabla h_i\|_{L^2(\Omega_x)}^2 \left| W_{i,y}^1 y \right|^2}{\left\| \overline{W}_{i,x}^1 \right\|^2} + C_{d_x} \frac{\left| \overline{W}_i^2 \right|^2 \left| W_{i,y}^1 y \right|^3}{3 \left\| \overline{W}_{i,x}^1 \right\|},$$

where C_{d_x} denotes the measure of the largest $(d_x - 1)$-hyperplane in Ω_x and

$$h_i(x) = \overline{W}_i^2 \alpha(\overline{W}_{i,x}^1 x + \overline{b}_i^1), \quad f(x, 0) = \sum_{i=1}^{n_1} h_i(x) + \overline{b}^2.$$

Notice that $h_i(x)$ are Lipschitz in x so $||\nabla h_i||_{L^2(\Omega)}^2$ is well defined. We denote $h_i(x)$ explicitly and separately since $\nabla_x f(x, 0) = \sum_{i=1}^{n_1} \nabla h_i(x)$ where $f(x, 0)$ could be the approximation of the target function $g(x)$ on $\mathcal{M}$. The estimate presented in Lemma 3 is a type of

a posteriori estimate since it depends on the parameters $\overline{W}^{\ell}$ and $\bar{b}^2$ obtained as the results of training.

We first notice that the stability of each trained neuron depends on the derivative of h with respect to each input variable. The derivatives depend on the trained parameters that are directly connected to the input vector. These parameters depend on the data and the training algorithm. Furthermore, we observe that the stability of a neuron is dependent on the "untrainable" parameters in W_y^1! Finally, the lemma suggests that if the trained network is more stable if the weight W_i^2 connecting to the output is small. This matches with our intuition that W_i^2 may amplify the contribution of the y components of the input. By summing all $e_i(x, y)$ together and applying the triangle inequality, we have the following estimate for trained ReLU DNNs with one hidden layer.

Theorem 5 *Let $\boldsymbol{x}_i \sim M_0$ in the training data and $f(x, y)$ be a network with a single hidden layer ($L = 2$) defined in (3.4) and trained by FGD or SGD, then*

$$\left\|f(\cdot, y) - f(\cdot, 0)\right\|_{L^2(\Omega_x)}^2 \leq \sum_{i=1}^{n_1} \left(\frac{\left|W_{i,y}^1 y\right|^2 \left\|\nabla h_i\right\|_{L^2(\Omega_x)}^2}{\left\|\overline{W}_{i,x}^1\right\|^2} + C_{d_x} \frac{\left|\overline{W}_i^2\right|^2 \left|W_{i,y}^1 y\right|^3}{3\left\|\overline{W}_{i,x}^1\right\|} \right),$$

where C_{d_x} and $h_i(x)$ follow the same definitions in Lemma 3.

This theorem gives the stability estimate for a ReLU DNN with one hidden layer trained by FGD or SGD. It is the building block for understanding the stability of a deep neural network. The next step is to use the nonlinear recursion relations that define the deep network to propagate the influence of having nonzero y components in the input vector input the other hidden layers.

3.2 Stability estimate for $L > 2$

For a general multilayer neural network with ReLU activation function, as shown in (3.1), we denote the function trained by FGD or SGD as $f(\boldsymbol{x}; \theta) = f^L(\boldsymbol{x})$ where

$$f^{\ell}(\boldsymbol{x}) = \overline{W}^{\ell} \alpha(f^{\ell-1}(\boldsymbol{x})) + \bar{b}^{\ell}, \quad \ell = 2 : L,$$

with $f^1(\boldsymbol{x}) = \overline{W}^1 \boldsymbol{x} + \bar{b}^1$. Let $f^{\ell}(\boldsymbol{x})$, $\ell = 1, \cdots, L$ be the functions in (3.1) and

$$\triangle_y f^{\ell}(x, y) := f^{\ell}(x, y) - f^{\ell}(x, 0).$$

In particular,

$$\triangle_y f(x, y) := f^L(x, y) - f^L(x, 0).$$

We have the following recursion relation of $\triangle_y f^{\ell}(x, y)$.

Lemma 4 *For any fixed $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$, we have*

$$\left\|\triangle_y f^{\ell}(x, y)\right\| \leq \left\|\overline{W}^{\ell}\right\| \left\|\triangle_y f^{\ell-1}(x, y)\right\|,$$

where $\left\|\triangle_y f^{\ell}(x, y))\right\|$ denotes the ℓ^2 vector norm of $\triangle_y f^{\ell}(x, y)$ and $\left\|\overline{W}^{\ell}\right\|$ is the operator norm of $\overline{W}^{\ell}$ with respect to ℓ^2 norm.

Proof By definition,

$$\left\|\triangle_y f^{\ell}(x, y)\right\|^2 = \left\|\overline{W}^{\ell} \left(\alpha\left(f^{\ell-1}(x, y)\right) - \alpha\left(f^{\ell-1}(x, 0)\right)\right)\right\|^2$$

$$
\begin{aligned}
&= \left\| \overline{W}^{\ell} \left(\alpha \left(f^{\ell-1}(x,0) + \triangle_y f^{\ell-1}(x,y) \right) - \alpha \left(f^{\ell-1}(x,0) \right) \right) \right\|^2 \\
&\le \left\| \overline{W}^{\ell} \right\|^2 \left\| \left(\alpha \left(f^{\ell-1}(x,0) + \triangle_y f^{\ell-1}(x,y) \right) - \alpha \left(f^{\ell-1}(x,0) \right) \right) \right\|^2 \\
&\le \left\| \overline{W}^{\ell} \right\|^2 \left\| \triangle_y f^{\ell-1}(x,y) \right\|^2 .
\end{aligned}
$$

The last inequality holds because of the property of ReLU that $|\mathrm{ReLU}(x+h) - \mathrm{ReLU}(x)| \le |h|$ for any $x, h \in \mathbb{R}$. □

By applying the previous recursion result, we have

$$
\left\| \triangle_y f^{\ell}(x,y) \right\| \le \left\| \overline{W}^{\ell} \right\| \left\| \triangle_y f^{\ell-1}(x,y) \right\| \le \cdots \le \left(\prod_{j=3}^{\ell} \left\| \overline{W}^{j} \right\| \right) \left\| \triangle_y f^2(x,y) \right\|
$$

Combining Lemmas 3 and 4, we have the following *a posteriori* estimate for $\left\| \triangle_y f(x,y) \right\|^2_{L^2(\Omega_x)}$.

Theorem 6 *Let $\boldsymbol{x}_i \sim M_0$ in the training data and $f(x,y)$ be a network with L layers defined in (3.4), then the following inequality holds for any fixed $y \in \mathbb{R}^{d_y}$ if $f(x,y)$ is trained by FGD or SGD:*

$$
\left\| \triangle_y f(\cdot,y) \right\|^2_{L^2(\Omega_x)} \le \left(\prod_{\ell=3}^{L} \left\| \overline{W}^{\ell} \right\|^2 \right) \sum_{\substack{i=1:n_2 \\ j=1:n_1}} \left(\frac{\left| W^1_{j,y} y \right|^2 \left\| \nabla_x h_{i,j} \right\|^2_{L^2(\Omega)}}{\left\| \overline{W}^1_{j,x} \right\|^2} + C_{d_x} \frac{\left| \overline{W}^2_{i,j} \right|^2 \left| W^1_{j,y} y \right|^3}{3 \left\| \overline{W}^1_{j,x} \right\|} \right), \tag{3.6}
$$

where C_{d_x} follows the definition in Lemma 3 and

$$
h_{i,j} = \overline{W}^2_{i,j} \alpha(\overline{W}^1_{j,x} x + \overline{b}^1_j).
$$

Proof By definition, we have

$$
\begin{aligned}
\left\| \triangle_y f(\cdot,y) \right\|^2_{L^2(\Omega_x)} &\equiv \left\| \triangle_y f^L(\cdot,y) \right\|^2_{L^2(\Omega_x)} \le \left(\prod_{\ell=3}^{L} \left\| \overline{W}^{\ell} \right\|^2 \right) \left\| \triangle_y f^2(\cdot,y) \right\|^2_{L^2(\Omega_x)} \\
&\le \left(\prod_{\ell=3}^{L} \left\| \overline{W}^{\ell} \right\|^2 \right) \sum_{i=1}^{n_2} \left\| \sum_{j=1}^{n_1} \overline{W}^2_{i,j} \left(\alpha(\overline{W}^1_{j,x} x + \overline{b}^1_j + W^1_{j,y} y) - \alpha(\overline{W}^1_{j,x} x + \overline{b}^1_j) \right) \right\|^2_{L^2(\Omega_x)} \\
&\le \left(\prod_{\ell=3}^{L} \left\| \overline{W}^{\ell} \right\|^2 \right) \sum_{i=1}^{n_2} \sum_{j=1}^{n_1} \left(\frac{\left| W^1_{j,y} y \right|^2 \left\| \nabla_x h_{i,j} \right\|^2_{L^2(\Omega)}}{\left\| \overline{W}^1_{j,x} \right\|^2} + C_{d_x} \frac{\left| \overline{W}^2_{i,j} \right|^2 \left| W^1_{j,y} y \right|^3}{3 \left\| \overline{W}^1_{j,x} \right\|} \right).
\end{aligned}
$$

□

Theorem 6 provides an estimation for the variation of a ReLU DNN trained by FGD or SDG along the normal direction of the data manifold. It is by no means sharp, because of the approximation (B.3). However, as in the case of LNNs, the initialization of W^1_y and the network's depth L play a role in the stability of the trained network as shown in Corollary 2. Theorem 6 has an interesting implication for DNNs that employ a latent space of a smaller dimensionality. The estimate in the theorem does not assume that the hidden layers in the DNN have the same width. This means that when $y \neq 0$, the effect of the "untrainable parameters" $W^1_y y$ will propagate into the subsequent layers, even when the layers have

smaller widths. In training for data without noise, $y \equiv 0$, there is no mechanism to learn how to project $W_y^1 \tilde{y}$ out for any $\tilde{y} \neq 0$ in noise test data. As far as we know, this is the first stability estimate ($\left\| \triangle_y f(\cdot, y) \right\|^2_{L^2(\Omega_x)}$) for a general ReLU DNN trained by FGD or SGD.

Corollary 2 *Under the same assumptions in Theorem 6 and*

$$D(y) := \max_{i,j,k} \left\{ \frac{[y]_k^2 \|\nabla_x h_{i,j}\|^2_{L^2(\Omega)}}{\left\| \overline{W}^1_{j,x} \right\|^2}, C_{d_x} \frac{[y]_k^3 \left| \overline{W}^2_{i,j} \right|^2}{3 \left\| \overline{W}^1_{j,x} \right\|^2} \right\},$$

if $\left[W^1_{j,y} \right]_k \sim \mathcal{N}(0, \nu^2)$ *for all* $k = 1 : d_y$, *then there exists a constant* $\widetilde{D}$ *such that*

$$\left\| \triangle_y f(\cdot, y) \right\|^2_{L^2(\Omega_x)} \leq \left(\prod_{\ell=3}^{L} \left\| \overline{W}^\ell \right\|^2 \right) \left(\left(\nu^2 + 2\sqrt{\frac{2}{\pi}} \nu^3 \right) n_2 n_1 d_y + \widetilde{D} \sqrt{n_2 n_1 d_y} \right) D(y), \tag{3.7}$$

with high probability. Here, n_1 *and* n_2 *are the widths of the first and second hidden neuron layers defined in* (3.1).

Commonly used initialization strategies correspond to $\nu^2 = \frac{1}{d}$ in [23] or $\nu^2 = \frac{2}{d+n_1}$ in [28]. Recently, the authors in [13] propose to take $\nu^2 = \frac{2}{\sqrt{n_1 d}}$ which leads to the following estimate:

$$\left\| \triangle_y f(\cdot, y) \right\|^2_{L^2(\Omega_x)} \leq \overline{D} \left(\prod_{\ell=3}^{L} \left\| \overline{W}^\ell \right\|^2 \right) n_2 \sqrt{n_1 d_y} D(y),$$

where $\overline{D} = \max\{4, \widetilde{D}\}$.

The above corollary suggests that, in additional to the common practice, the width of the second hidden layer should be considered in the initialization of $W^1_{j,y}$.

For classification problems, our theory provides additional understanding of adversarial examples [24]. Particularly, our theory may explain the existence of those adversarial examples which are close to the training examples according to some norm defined on the ambient space but are not a member of some idealized lower-dimensional data manifold. The estimate in (3.6) indicates that the variation of a trained ReLU DNN can significantly move the "decision boundary" for a small y provided $\left\| \overline{W}^\ell \right\|$ or $\left\| \nabla_x h_{i,j}(x) \right\|^2_{L^2(\Omega)}$ are sufficiently large. In this case, one can obtain adversarial examples easily with a very small perturbation along the normal direction of the data manifold. In addition, this result combined with the second conclusion in Lemma 2 and numerical results in Fig. 11 indicate that including a "weight decay" term in the loss function may reduce the reliability of a ReLU DNN-based classifier, as least when the data manifold is nearly flat.

In this section, we focused on estimating the stability of ReLU neural networks trained by FGD or SGD. Theorems 5 and 6 reveal the influence of the "trainable" and "non-trainable" parameters, $\overline{W}^\ell$ (including $\overline{W}^1_x$) and W^1_y, to the inference stability. The influence of the non-trainable parameters is unchanged, even if $\overline{W}^\ell$ are replaced by non-optimal ones. However, if the target functions fall into those considered in [17], W^1_y will be trainable, and the theoretical optimal inference error derived there is applicable.

We present some numerical results in the following subsection to demonstrate the above estimates. In particular, the stability metrics of ReLU DNNs with one hidden layer ($L = 2$) may differ from multi-hidden-layer ($L > 2$) cases since the product term will disappear if $L = 2$. This is observed in Fig. 10.

3.3 Numerical experiments

In this section, we present a series of numerical examples demonstrating the theorems presented in this paper.

The setup We take $d_x = 3$ and $d_y = 2$, i.e., $x \in \mathbb{R}^3$, $y \in \mathbb{R}^2$ and $\boldsymbol{x} = (x, 0)$. A total of 5×10^3 training data points generated by sampling $g_i = g(x_i) = \sum_{j=1}^{3} \sin\left(\pi [x_i]_j\right)$ with $x_i \sim U\left([-1, 1]^{d_x}\right)$. The hidden layers in a network have the same with, denoted by n.

The ReLU DNNs and their optimization are implemented using PyTorch [43].

The networks are trained for 100 epochs by using SGD without momentum or weight decay. The mini-batch size is chosen as 50, and the learning rate decays from 10^{-2} to 10^{-4} under a cosine annealing schedule [37].

To compute the stability estimates, we adopt the Monte Carlo approximation

$$\mathbb{E}_y\left[\left\|\triangle_y f(\cdot, y)\right\|^2_{L^2(\Omega_x)}\right] \approx \frac{1}{M}\sum_{i=1}^{M}\left(f(x_i, y_i) - f(x_i, 0)\right)^2, \tag{3.8}$$

where $x_i \sim U\left([-1, 1]^{d_x}\right)$ and $y_i \sim N(0, \gamma^2 I_{d_y})$. The weights W_y^1 are initialized following the special form:

$$W^1_{i,y} = \eta(1, 1)^T.$$

All other weights are initialized according to [28]. We take $M = 5 \times 10^3$ to evaluate the stability metric, and the final results are obtained by averaging 10 individual tests.

Numerical confirmation of various rates Theorems 5 and 6 state that with a fixed weight set W_y^1,

$$\mathbb{E}_y\left[\left\|\triangle_y f(\cdot, y)\right\|^2_{L^2(\Omega_x)}\right] \sim \mathcal{O}\left(\gamma^2\right),$$

if $y \sim \mathcal{N}(0, \gamma^2 I_{d_y})$. On the other hand,

$$\mathbb{E}_y\left[\left\|\triangle_y f(\cdot, y)\right\|^2_{L^2(\Omega_x)}\right] \sim \mathcal{O}\left(\eta^2\right)$$

if $W^1_{i,y}$ is initialized $\eta(1, 1)$ and $\gamma = 1$ in the distribution of y. Figure 10 demonstrates such scalings for networks of different depths. Also from Fig. 10 one may observe a gap between the curve from $L = 2$ and those $L > 2$. This gap seems to suggest that ReLU DNNs with one hidden layer differ from multi-hidden-layer models. Results in Fig. 10 further support this observation if we compare with some deeper ReLU DNNs. This phenomenon can be partially interpreted as the effect of the term $\prod_{\ell=3}^{L}\left\|\overline{W}^{\ell}\right\|^2$ as shown in Theorem 6.

Regularization by adding a "weight decay" term We recall the second statement in Lemma 2 that the ℓ^2 regularization term $\lambda\|\theta\|^2_{\ell^2}$ will significantly affect the stability factor

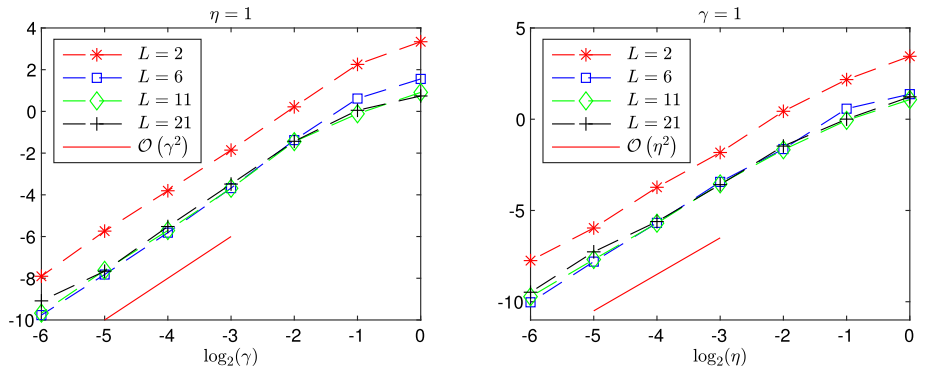

Fig. 10 Plots of $\log_2\left(\mathbb{E}_y\left[\|\triangle_y f(\cdot,y)\|^2_{L^2(\Omega_x)}\right]\right)$, with $y \sim \mathcal{N}(0,\gamma^2 I_{d_y})$ and $W^1_y = \eta(1,1)^T$. Each hidden layer of the networks has $n = 100$ neurons. The plots verify the estimate Theorem 6

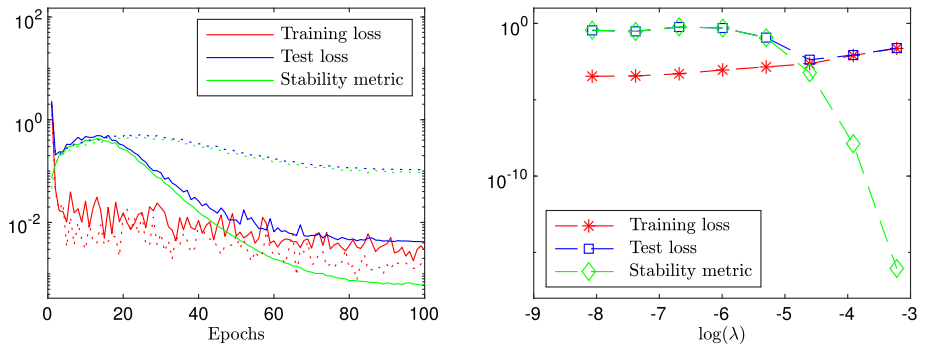

Fig. 11 Effect of adding the "weight decay" term $\lambda\|\theta\|^2_{\ell^2}$ in the total loss function. Training loss, test loss, and stability metric $\mathbb{E}_y\left[\|\triangle_y f(\cdot,y)\|^2_{L^2(\Omega_x)}\right]$ in training process (left) and their final results (right). In training process (left), the dashed lines represent the results of $\lambda = 5\times 10^{-3}$ and the solid lines represent $\lambda = 10^{-2}$. The final results (right), shows that trade-off between the model accuracy and stability metric

$\mathbb{E}_y\left[\|\triangle_y f(\cdot,y)\|^2_{L^2(\Omega_x)}\right]$. Thus, we show the training process and final loss (training loss, test loss and stability metric) with different values of λ in Fig. 11.

In this example, the training loss is defined in (3.2) with $y \equiv 0$ and test loss is calculated with the same formula while it shares the same sampled date points in computing $\mathbb{E}_y\left[\|\triangle_y f(\cdot,y)\|^2_{L^2(\Omega_x)}\right]$ with $y \sim \mathcal{N}(0,\gamma^2 I_{d_y})$ and $\gamma = 2^{-1}$, and the initialization of $W^1_{i,y}$ is $(1,1)$, i.e., $\eta = 1$.

This example shows that (i) there is no surprise that regularizing the ℓ^2 norm of the weight set reduces the stability metric $\mathbb{E}_y\left[\|\triangle_y f(\cdot,y)\|^2_{L^2(\Omega_x)}\right]$; however, (ii) both the training and test losses will increase as the magnitude of the regularization, λ, increases. In practice, a suitable scale of λ is critical to balance the approximation error and the regularization effect for the stability metric $\mathbb{E}_y\left[\|\triangle_y f(\cdot,y)\|^2_{L^2(\Omega_x)}\right]$.

Regularization by introducing noise to the data Motivated by the analysis for LNNs in Sect. 2.4.1, we study numerically the potential of stabilization by adding noise to the data set. We follow the setup introduced above, except that we have noisy data $\boldsymbol{x}_i \sim M_\sigma$, i.e.,

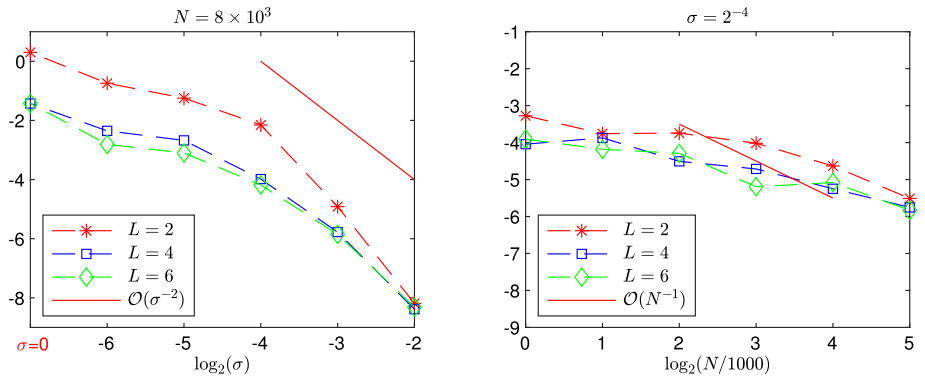

Fig. 12 $\log_2\left(\mathbb{E}_y\left[\left\|\Delta_y f(\cdot,y)\right\|^2_{L^2(\Omega_x)}\right]\right)$ as a function of σ, the variance in the noise added to the training data (left). $\log_2\left(\mathbb{E}_y\left[\left\|\Delta_y f(\cdot,y)\right\|^2_{L^2(\Omega_x)}\right]\right)$ as a function of the size of training data set, N (right)

$\boldsymbol{x}_i = \begin{pmatrix} x_i \\ \sigma y_i \end{pmatrix}$, where $x_i \sim X$ and $y_i \sim N(0, I_{d_y})$. In addition, we take $\eta = 1$, i.e., $W^1_{i,y} = (1, 1)$, to initialize $W^1_{i,y}$.

The stability metric $\mathbb{E}_y\left[\left\|\Delta_y f(\cdot,y)\right\|^2_{L^2(\Omega_x)}\right]$ is evaluated with $y \sim N(0, \gamma I_{d_y})$, $\gamma = 2^{-2}$, and approximated by summation of $M = 8 \times 10^3$ independent samples for the case of comparing different σ (different level of added noise) and $M = 4 \times 10^4$ samples for the case of comparing training sets of different cardinality, N.

The curves shown in Fig. 12 are obtained by averaging 5 individual tests. Figure 12 verifies our conjectures about stabilization effect of noise to the normal direction of data manifold and increasing the data points. More discussion about these results will be presented in the following section.

4 Stability from adding noise to the data manifold

In this section, we consider on a more abstract level the effects of adding noise to the embedded low-dimensional data. The aim is to improve the trained neural network's stability, evaluating points that lie out of the training data distribution. We have seen in the previous sections that adding noise may regularize the optimization problem in some sense and provide stability. In the following we shall relate adding noise in the normal directions of the given data manifold to implicitly defining an extension of the loss function (1.7). The change in the loss function subsequently enables the learning function to approximate the constant normal extension, $\overline{g}$ as defined in (1.9), of the label function g. This view provides a more intuitive explanation of how adding noise according to the geometry of the data may enhance the stability of a trained network, provided that the data set is sufficiently large.

4.1 Implicit extension of the loss functional

Let $\mathcal{M}$ be a d_x-dimensional compact C^2-manifold in $\mathbb{R}^d$. Denote by $N_{\boldsymbol{x}}M$ the normal space of $\mathcal{M}$ at $\boldsymbol{x} \in \mathcal{M}$ and $r > 0$ the reach of $\mathcal{M}$. For any $\sigma \in (0, r)$, we introduce the σ-tubular neighborhood of $\mathcal{M}$ in $\mathbb{R}^d$ as

$$T_\sigma := \left\{\boldsymbol{x} + \epsilon \boldsymbol{n}_x \ : \ \boldsymbol{x} \in \mathcal{M}, \epsilon \in (-\sigma, \sigma), \boldsymbol{n}_x \in N_{\boldsymbol{x}}M, \text{ and } \|\boldsymbol{n}_x\| = 1\right\}.$$

For points in T_σ, define the projection

$$\mathcal{P}_{\mathcal{M}}\boldsymbol{x} = \arg\inf_{\boldsymbol{\xi}\in\mathcal{M}} \|\boldsymbol{x}-\boldsymbol{\xi}\|.$$

Now, let $U[\mathcal{M}]$ denote the uniform distribution defined on $\mathcal{M}$; i.e., the density of $U[\mathcal{M}]$ is uniform with respect to the measure on $\mathcal{M}$, induced by the Euclidean norm of $\mathbb{R}^d$. To each data point $\boldsymbol{x}$ sampled independently from $U[\mathcal{M}]$, we introduce noise that lifts $\boldsymbol{x}$ to $\widetilde{\boldsymbol{x}}$ in the normal space $N_{\boldsymbol{x}}M$. More precisely,

$$\widetilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \boldsymbol{n}_x,$$

where $\boldsymbol{n}_x \in N_{\boldsymbol{x}}M$ is sampled from the uniform distribution on $\mathbb{S}^{d_y-1}$ embedded in $N_{\boldsymbol{x}}M$ and $\epsilon \sim U[-\sigma,\sigma]$ with $\sigma < r$. We shall denote the resulting joint distribution as M_σ and its density ρ_σ. Thus, $\tilde{\boldsymbol{x}}$ is a point in T_σ, sampled from M_σ. According to the coarea formula, ρ_σ is uniform on T_σ only if $\mathcal{M}$ is flat. See [15,35] for the case when $\mathcal{M}$ is a hypersurface.

The loss function defined with the noisy data $\left\{(\widetilde{\boldsymbol{x}}_i, g_i)\right\}_{i=1}^N$ $(g_i = g(\boldsymbol{x}_i))$ can be written as

$$J(\theta) = \frac{1}{2N}\sum_{i=1}^N \left|f_\theta(\widetilde{\boldsymbol{x}}_i) - g_i\right|^2 = \frac{1}{2N}\sum_{i=1}^N \left|f_\theta(\widetilde{\boldsymbol{x}}_i) - g(\mathcal{P}_{\mathcal{M}}\widetilde{x}_i)\right|^2. \tag{4.1}$$

In other words, J can be interpreted as the empirical loss of the following continuous loss

$$\overline{\mathcal{J}}(\theta) := \frac{1}{2}\int_{T_\sigma} \left|f_\theta(\boldsymbol{x}) - \overline{g}(\boldsymbol{x})\right|^2 \rho_\sigma(\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \tag{4.2}$$

where $\overline{g}(\boldsymbol{x}) := g\left(\mathcal{P}_{\mathcal{M}}\boldsymbol{x}\right)$ is the constant extension of $g(\boldsymbol{x})$ along the normal directions. This implies that the "regularization" effect from using this type of noisy data is the "automatic learning" of $\overline{g}$ on $\mathbb{R}^d$.

4.2 Accuracy/stability trade-off

Assuming that we do not know the geometry of the data manifold, we add noise to every component in the ambient space indifferently.

For simplicity, we assume the data set $D_N = \{(\boldsymbol{x}_i, g_i)\}_{i=1}^N$ consists of

$$\boldsymbol{x}_i = \begin{pmatrix} x_i \\ 0 \end{pmatrix} + \sigma \begin{pmatrix} \epsilon_{i,x} \\ \epsilon_{i,y} \end{pmatrix} \in \mathbb{R}^{d_x+d_y},$$

where $(\epsilon_{i,x}, \epsilon_{i,y}) = \boldsymbol{\epsilon}_i \sim N(0, I_{d_x+d_y})$ sampled as the noise part. In addition, the "label" in the data are clean and followed by $g_i = g(x_i)$ for every $x_i \in \mathbb{R}^{d_x}$.

Thus, we have

$$g_i = g(x_i + \sigma\epsilon_{i,x} - \sigma\epsilon_{i,x}) = g(x_i + \sigma\epsilon_{i,x}) + \mathcal{O}(\sigma).$$

This means we can interpret the noisy data as

$$(\boldsymbol{x}_i, g_i) = \left(\begin{pmatrix} x_i + \sigma\epsilon_{i,x} \\ \sigma\epsilon_{i,y} \end{pmatrix}, g(x_i)\right) = \left(\begin{pmatrix} \tilde{x}_i \\ \sigma\epsilon_{i,y} \end{pmatrix}, g(\tilde{x}_i) + \mathcal{O}(\sigma)\right), \tag{4.3}$$

where $\tilde{x}_i = x_i + \sigma\epsilon_{i,x}$. Thus, for any trained machine learning model $f(x,y)$, we can decompose the generalization error as

$$\left\|f(x,y) - gx)\right\|^2 \le \left\|f(x,0) - g(x)\right\|^2 + \left\|f(x,y) - f(x,0)\right\|^2.$$

Table 1 Linear regression results of β with different L and co-dimension d_y

d_y	1	2	3	4	5	6	7	8	9	10
$L=2$	0.40	0.40	0.35	0.36	0.35	0.39	0.38	0.33	0.38	0.37
$L=4$	0.42	0.37	0.40	0.34	0.43	0.39	0.42	0.43	0.43	0.41
$L=6$	0.46	0.38	0.40	0.44	0.44	0.41	0.43	0.46	0.46	0.46

The interpolation error $\|f(x,0)-g(x)\|^2$ corresponds to the error for the classical learning task with noisy label data $(\tilde{x}_i, \tilde{g}_i)$ where $\tilde{x}_i = x_i + \sigma\epsilon_{i,x} \sim \widetilde{X} := X + \sigma N(0, I_{d_x}) \in R^{d_x}$ and $\tilde{g}_i = g(\tilde{x}_i) + \mathcal{O}(\sigma) \in \mathbb{R}$. Then, it follows that

$$\begin{aligned}\mathbb{E}_{x\sim X}\left[\|f(x,0)-g(x)\|^2\right] &\le \mathbb{E}_{\tilde{x}\sim\widetilde{X}}\left[\|f(\tilde{x},0)-g(\tilde{x})\|^2\right] + \mathcal{O}(\sigma^2)\\ &= \mathbb{E}_{\tilde{x}\sim\widetilde{X}}\left[\|f(\tilde{x},0)-\tilde{g}(\tilde{x})+\mathcal{O}(\sigma)\|^2\right] + \mathcal{O}(\sigma^2)\\ &\le \langle\|f(\tilde{x},0)-\tilde{g}\|^2\rangle_N + \mathcal{O}\left(\sigma^2\right) + \mathcal{O}\left(N^{-1}\right).\end{aligned} \tag{4.4}$$

Here, $\langle\|f(\tilde{x},0)-\tilde{g}\|^2\rangle_N$ is the empirical loss which can be bounded by the approximation power of one-hidden-layer ($L=2$) neural networks [7,50] and deep ($L>2$) neural networks [49,57].

The stability metric, by which we mean $\left\|\triangle_y f(x,y)\right\| = \|f(x,y)-f(x,0)\|^2$, for LNNs is estimated to be $\mathcal{O}\left(\left(\sigma^2 N\right)^{-1}\right)$. If $\epsilon_{i,x}=0$, the reciprocal relation between $\left\|\triangle_y f(x,y)\right\|^2$ and variance σ^2 is observed in ReLU DNNs in Fig. 12. However, Fig. 12 suggests that $\left\|\triangle_y f(x,y)\right\|^2$ for ReLU DNNs is reciprocal to N^β with $\beta<1$, in contrast to $\beta=1$ in the case of LNNs.

We present Table 1, which summarizes a series of further numerical experiments and reveals how β is related to network's depth and the codimensions, d_y, of the data manifold. In the table, β is fitted by using the linear regression for $\|\triangle_y f(x,y)\|^2$ and N in the logarithmic scale.

From the table, we find that

$$\left\|\triangle_y f(x,y)\right\|^2 \approx \mathcal{O}\left(\frac{1}{\sigma^2 N^\beta}\right), \tag{4.5}$$

where $\beta < 1/2$ seems to relate to the depth of the network, but independent of the co-dimension of the data manifold. Again, the experimental results are quite different from LNN case. The results suggest that nonlinear ReLU networks require more training data to control the variation of the neural networks in the y-directions (for small σ).

For any fixed data set (fixed N), 4.4 and 4.5 describe a trade-off between accuracy and stability: On the one hand, reducing the fitting errors in (4.4) requires smaller noise level for the x-components. On the other hand, small noise level in the y-direction will decrease the stability of f in the y-direction. However, if the data manifold is not flat, the geometry of the manifold will impose an additional constraint to the maximal noise level. Too large of a noise level will lead to ill-conditioned optimization problem.

5 Summary

Surprising features in supervised learning problems arise when data are embedded in a high-dimensional Euclidean space. We derived estimates on the derivatives of the learning function in the direction transversal to the data subspace. When a neural network defines the learning function, a portion of its weights is untrainable by a typical gradient descent-based algorithm because the empirical loss function is independent of these weights.

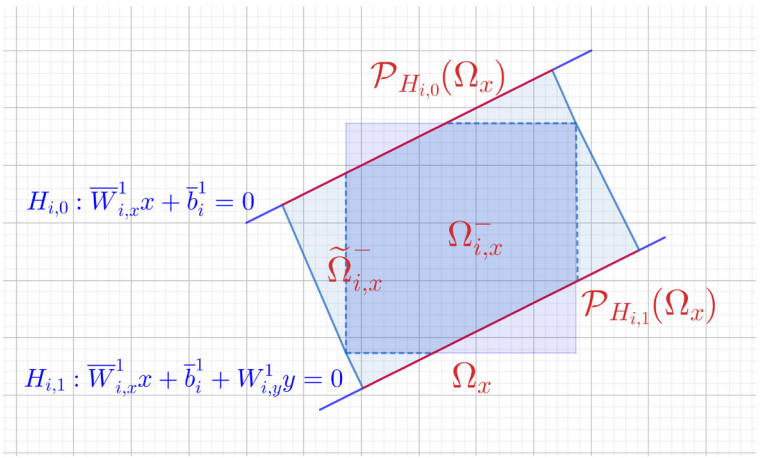

Fig. 13 Diagram of $\Omega^-_{i,x}$ and $\widetilde{\Omega}^-_{i,x}$

Consequently, the learning function's values at points away from the data subspace depend on the initialization of the untrainable weights.

We showed that if noise in the codimension of the data subspace is present, the weights in question can be controlled, provided that the training data size is sufficiently large. However, the training data size only has to be large compared with the standard deviation σ and seems independent of the number of codimensions. For linear networks, we have shown that the price for this regularization is the slow convergence for those weights to small numbers. We have also demonstrated that the network's depth may provide a particular regularization effect if the network's weights are initialized in a suitable subregion of $\mathbb{R}^d$. For nonlinear networks activated by ReLU, similar to LNNs, there is still a set of parameters that are not trainable if the data subspace has nonzero number of codimensions. We derived a stability estimate for the influence of the untrainable weights in a trained neural network.

Though adding noise to the data set may provide a desired regularization to the learning function, it also incurs a trade-off to the accuracy of the trained network and possibly renders the optimization model ill-conditioned, when the data manifold is not flat. It is also clear that if one has more information about the geometry of the data manifold, one can introduce noise adaptively according to the manifold's geometry and mitigate the loss of accuracy.

Acknowledgements
The authors thank Lukas Taus for his help with the numerical experiments in Fig. 9. Tsai's research is supported partially by the National Science Foundation Grants DMS-2110895 and by Army Research Office, under Cooperative Agreement Number W911NF-19-2-0333. Ward's research is supported in part by AFOSR MURI FA9550-19-1-0005, NSF DMS 1952735, NSF HDR-1934932, and NSF 2019844. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the US Government. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Data availability statement
The authors declare that the data supporting the findings of this study are available within the article.

Author details
[1]Department of Mathematics, The University of Texas at Austin, Austin, TX 78712, USA, [2]Department of Mathematics and Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, USA.

A Proofs for LNNs

A.1 Proof of Proposition 1

Proof We first show that the loss function can be transformed naturally under the unitary mapping Q. The original loss function can be formulated as

$$J^e(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\boldsymbol{w}^T Q\begin{pmatrix} x_i \\ \sigma y_i \end{pmatrix} - g_i)^2 = \frac{1}{2N}\sum_{i=1}^{N}((Q^T\boldsymbol{w})^T\begin{pmatrix} x_i \\ \sigma y_i \end{pmatrix} - g_i)^2.$$

Thus, if we denote

$$\widetilde{\boldsymbol{w}} = Q^T\boldsymbol{w},$$

we can define the new loss function $\widetilde{J}^e(\widetilde{\boldsymbol{w}})$ with respect to the new variable $\widetilde{\boldsymbol{w}}$ as

$$\widetilde{J}^e(\widetilde{\boldsymbol{w}}) = J^e(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=1}^{N}(\widetilde{\boldsymbol{w}}^T\begin{pmatrix} x_i \\ \sigma y_i \end{pmatrix} - g_i)^2.$$

In addition, by taking the gradient for $J^e(\boldsymbol{w})$ with respect to $\boldsymbol{w}$, we have

$$\nabla_{\boldsymbol{w}} J^e(\boldsymbol{w}) = Q\nabla_{\widetilde{\boldsymbol{w}}}\widetilde{J}^e(\widetilde{\boldsymbol{w}}).$$

Furthermore, we claim that the dynamic system for w can be rotated to $\widetilde{\boldsymbol{w}}$ naturally. First, we can check

$$\mathcal{P}_{\boldsymbol{w}}(\boldsymbol{v}) = \frac{\boldsymbol{w}\boldsymbol{w}^T}{\|\boldsymbol{w}\|^2}\boldsymbol{v} = \frac{Q\widetilde{\boldsymbol{w}}\widetilde{\boldsymbol{w}}^T Q^T}{\|\widetilde{\boldsymbol{w}}\|^2}\boldsymbol{v} = Q\mathcal{P}_{\widetilde{\boldsymbol{w}}}(Q^T\boldsymbol{v}).$$

Based on the dynamical system for w, we have

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} &= -\|\boldsymbol{w}\|^{2-\frac{2}{L}}\left(\nabla_{\boldsymbol{w}}J^e(\boldsymbol{w}) + (L-1)\mathcal{P}_{\boldsymbol{w}}(\nabla_{\boldsymbol{w}}J^e(\boldsymbol{w}))\right)\\ &= -\|\widetilde{\boldsymbol{w}}\|^{2-\frac{2}{L}}\left(Q\nabla_{\widetilde{\boldsymbol{w}}}\widetilde{J}^e(\widetilde{\boldsymbol{w}}) + (L-1)Q\mathcal{P}_{\widetilde{\boldsymbol{w}}}(Q^TQ\nabla_{\widetilde{\boldsymbol{w}}}\widetilde{J}^e(\widetilde{\boldsymbol{w}}))\right)\\ &= -\|\widetilde{\boldsymbol{w}}\|^{2-\frac{2}{L}}Q\left(\nabla_{\widetilde{\boldsymbol{w}}}\widetilde{J}^e(\widetilde{\boldsymbol{w}}) + (L-1)\mathcal{P}_{\widetilde{\boldsymbol{w}}}(\nabla_{\widetilde{\boldsymbol{w}}}\widetilde{J}^e(\widetilde{\boldsymbol{w}}))\right).\end{aligned}$$

Finally, we can see that

$$\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\boldsymbol{w}} = Q^T\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} = -\|\widetilde{\boldsymbol{w}}\|^{2-\frac{2}{L}}\left(\nabla_{\widetilde{\boldsymbol{w}}}\widetilde{J}^e(\widetilde{\boldsymbol{w}}) + (L-1)\mathcal{P}_{\widetilde{\boldsymbol{w}}}(\nabla_{\widetilde{\boldsymbol{w}}}\widetilde{J}^e(\widetilde{\boldsymbol{w}}))\right).$$

□

A.2 Proof of Proposition 2

Proof Since

$$\mathcal{P}_{\boldsymbol{w}}(\boldsymbol{v}) = \frac{\boldsymbol{w}^T\boldsymbol{v}}{\|\boldsymbol{w}\|^2}\boldsymbol{w} = \frac{\boldsymbol{w}\boldsymbol{w}^T}{\|\boldsymbol{w}\|^2}\boldsymbol{v},$$

we have

$$\begin{aligned}\boldsymbol{F}(\boldsymbol{w}) &= -\|\boldsymbol{w}\|^{-\frac{2}{L}}\left(\|\boldsymbol{w}\|^2\nabla J^e(\boldsymbol{w}) + (L-1)\boldsymbol{w}\boldsymbol{w}^T\nabla J^e(\boldsymbol{w})\right)\\ &= -\|\boldsymbol{w}\|^{-\frac{2}{L}}\left(\left(\|\boldsymbol{w}\|^2 I_d + (L-1)\boldsymbol{w}\boldsymbol{w}^T\right)\nabla J^e(\boldsymbol{w})\right)\\ &= -\|\boldsymbol{w}\|^{-\frac{2}{L}}\boldsymbol{M}\nabla J^e(\boldsymbol{w}),\end{aligned}$$

where $\boldsymbol{M} = \|\boldsymbol{w}\|^2 I_d + (L-1)\boldsymbol{w}\boldsymbol{w}^T \in \mathbb{R}^{d\times d}$ is a symmetric positive definite matrix if $\boldsymbol{w} \neq \boldsymbol{0}$. Thus, $\boldsymbol{F}(\boldsymbol{w}) = 0$ if and only if $\boldsymbol{w} = \boldsymbol{0}$ or $\nabla J^e(\boldsymbol{w}) = \boldsymbol{0}$.

If $L > 2$ and $J^e(\boldsymbol{w})$ is strictly convex, there is a unique $\boldsymbol{w}^* \neq 0$ (since $\langle \boldsymbol{x}g\rangle_N \neq 0$) such that $\nabla J^e(\boldsymbol{w}^*) = 0$ and the Hessian matrix $\nabla^2 J^e(\boldsymbol{w}^*)$ is a symmetric positive definite (SPD) matrix. Then, the Jacobian matrix of $\boldsymbol{F}(\boldsymbol{w})$ at $\boldsymbol{w}^*$ is

$$\nabla \boldsymbol{F}(\boldsymbol{w}^*) = -\|\boldsymbol{w}^*\|^{-\frac{2}{L}} \boldsymbol{M}(\boldsymbol{w}^*)\nabla^2 J^e(\boldsymbol{w}^*).$$

Given $\boldsymbol{w}^* \neq 0$ and both $\boldsymbol{M}(\boldsymbol{w}^*)$ and $\nabla^2 J^e(\boldsymbol{w}^*)$ are SPD, we notice that

$$\nabla \boldsymbol{F}(\boldsymbol{w}^*) \sim \boldsymbol{M}^{-\frac{1}{2}}(\boldsymbol{w}^*)\nabla \boldsymbol{F}(\boldsymbol{w}^*)\boldsymbol{M}^{\frac{1}{2}}(\boldsymbol{w}^*) = -\|\boldsymbol{w}^*\|^{-\frac{2}{L}} \boldsymbol{M}^{\frac{1}{2}}(\boldsymbol{w}^*)\nabla^2 J^e(\boldsymbol{w}^*)\boldsymbol{M}^{\frac{1}{2}}(\boldsymbol{w}^*),$$

which shows that all eigenvalues of $\nabla \boldsymbol{F}(\boldsymbol{w}^*)$ are negative. Since $L > 2$, the leading order of $\boldsymbol{F}(\boldsymbol{w})$ is $\mathcal{O}(\|\boldsymbol{w}\|^{2-\frac{2}{L}})$ which is continuously differentiable at $\boldsymbol{w} = \boldsymbol{0}$ with $\nabla \boldsymbol{F}(\boldsymbol{w}) = 0$. □

A.3 Proof of Corollary 1

Proof From formula 2.11 in Proposition 3, we have $\langle A\rangle_N \begin{pmatrix} w_x^* \\ \sigma w_y^* \end{pmatrix} = \begin{pmatrix} \langle gx\rangle_N \\ \langle gy\rangle_N \end{pmatrix}$, where (w_x^*, w_y^*) is the solution with respect to the target function $g(x)$. Given $g(x) = \tilde{g}(x) + \mu^T x$, we have

$$\langle A\rangle_N \begin{pmatrix} w_x^* \\ \sigma w_y^* \end{pmatrix} = \begin{pmatrix} \langle gx\rangle_N \\ \langle gy\rangle_N \end{pmatrix} = \begin{pmatrix} \langle \tilde{g}x\rangle_N \\ \langle \tilde{g}y\rangle_N \end{pmatrix} + \begin{pmatrix} \langle (\mu^T x)x\rangle_N \\ \langle (\mu^T x)y\rangle_N \end{pmatrix}.$$

Since

$$\langle A\rangle_N \begin{pmatrix} \mu \\ 0 \end{pmatrix} = \begin{pmatrix} \langle (\mu^T x)x\rangle_N \\ \langle (\mu^T x)y\rangle_N \end{pmatrix},$$

we have

$$\langle A\rangle_N \begin{pmatrix} w_x^* - \mu \\ \sigma w_y^* \end{pmatrix} = \begin{pmatrix} \langle \tilde{g}x\rangle_N \\ \langle \tilde{g}y\rangle_N \end{pmatrix}.$$

Recalling the block structure of $\langle A\rangle_N$ and applying $\langle A\rangle_N^{-1}$ on both sides of the above equation, we have

$$w_x^* - \mu = \widetilde{\Sigma}_X^{-1}\left(\langle \tilde{g}x\rangle_N + \langle xy^T\rangle_N S^{-1}\langle yx^T\rangle_N \Sigma_X^{-1}\langle \tilde{g}x\rangle_N - \langle xy^T\rangle_N S^{-1}\langle \tilde{g}y\rangle_N\right) \tag{A.1}$$

and

$$w_y^* = \sigma^{-1}S^{-1}\left(\langle \tilde{g}y\rangle_N - \langle yx^T\rangle_N \widetilde{\Sigma}_X^{-1}\langle \tilde{g}x\rangle_N\right). \tag{A.2}$$

Given $|\widetilde{g}(x)| \leq \delta$ and the estimates in Theorem 1, we have $\|\widetilde{\Sigma}_X^{-1}\|, \|S^{-1}\| \sim \mathcal{O}(1)$, $\|\langle \tilde{g}x\rangle_N\| \lesssim \delta$, $\|\langle \tilde{g}y\rangle_N\| \lesssim \frac{\delta}{\sqrt{N}}$, $\|\langle xy^T\rangle_N \widetilde{\Sigma}_X^{-1}\langle \tilde{g}x\rangle_N\| \lesssim \frac{\delta}{\sqrt{N}}$, $\|\langle yx^T\rangle_N \widetilde{\Sigma}_X^{-1}\langle \tilde{g}x\rangle_N\| \lesssim \frac{\delta}{\sqrt{N}}$, and $\|\langle xy^T\rangle_N S^{-1}\langle yx^T\rangle_N \Sigma_X^{-1}\langle \tilde{g}x\rangle_N\| \lesssim \frac{\delta}{N}$. Here, $\lesssim$ means that there is a constant which depends only on the distribution X and Y. Then, the results can be obtained by taking norm in (A.1) and (A.2) and then substituting the previous estimates. □

A.4 Proof of Theorem 3

Proof Let us denote

$$f_{\max} := \sup_{\boldsymbol{w}\in B_{\frac{\epsilon}{2}}(\boldsymbol{w}(0))} \left\|\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w}\right\|,$$

where $B_{\frac{\epsilon}{2}}(\boldsymbol{w}(0)) := \{\boldsymbol{w} \ : \ \|\boldsymbol{w} - \boldsymbol{w}(0)\| \leq \frac{\epsilon}{2}\}$ is the $\frac{\epsilon}{2}$-ball centered at $\boldsymbol{w}(0)$. Given the continuity of $\boldsymbol{w}(t)$ and the definition of $T_L(\epsilon)$, we have

$$\frac{\epsilon}{2} = \|\boldsymbol{w}(T_L(\epsilon)) - \boldsymbol{w}(0)\| = \left\|\int_0^{T_L(\epsilon)} \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w}\mathrm{d}t\right\| \leq f_{\max} T_L(\epsilon).$$

It follows that

$$T_L(\epsilon) \geq \frac{\epsilon}{2f_{\max}}.$$

For $f_{\max}$, we notice that

$$\begin{aligned}\left\| \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w} \right\| &= \|\boldsymbol{w}\|^{2-\frac{2}{L}} \left\| \nabla_{\boldsymbol{w}} J^e(\boldsymbol{w}) + (L-1)\mathcal{P}_{\boldsymbol{w}}(\nabla_{\boldsymbol{w}} J^e(\boldsymbol{w})) \right\| \\ &\leq L\|\boldsymbol{w}\|^{2-\frac{2}{L}} \left\| \nabla_{\boldsymbol{w}} J^e(\boldsymbol{w}) \right\| \\ &= L\|\boldsymbol{w}\|^{2-\frac{2}{L}} \left\| \langle A_\sigma \rangle_N \boldsymbol{w} - \langle g\boldsymbol{x} \rangle_N \right\| \\ &\leq L\|\boldsymbol{w}\|^{2-\frac{2}{L}} \left(C_2\|\boldsymbol{w}\| + \|\langle g\boldsymbol{x} \rangle_N\| \right).\end{aligned}$$

Since $\boldsymbol{w} \in B_{\frac{\epsilon}{2}}(\boldsymbol{w}(0))$, $\|\boldsymbol{w}(0)\| = \epsilon$, and $\epsilon << 1$, there exists C that depends on L, C_2, and $\|\langle g\boldsymbol{x} \rangle_N\| = \mathcal{O}(1)$ such that

$$L\|\boldsymbol{w}\|^{2-\frac{2}{L}} \left(C_2\|\boldsymbol{w}\| + \|\langle g\boldsymbol{x} \rangle_N\| \right) \leq \frac{1}{2C}\|\epsilon\|^{2-\frac{2}{L}}.$$

This means $f_{\max} \leq \frac{1}{2C}\|\epsilon\|^{2-\frac{2}{L}}$, which finished the proof. □

A.5 Proof of Proposition 5

Before we show the proof of Proposition 5, let us first present the following lemma.

Lemma 5 *Let $A \in \mathbb{R}^{d\times d}$ be a symmetric positive definite (SPD) matrix with $d \geq 2$ and assume $a_1 \geq a_2 \geq \cdots \geq a_d$ are the its eigenvalues. Then, we have*

$$w^T A u \geq \frac{a_d - a_1}{2}\|w\|\|u\|,$$

if $w, u \in \mathbb{R}^d$ and $w^T u = 0$.

Proof First, we may assume the SVD decomposition for A as $A = V^T \Sigma V$, where V is a unitary matrix and $\Sigma = \mathrm{diag}(a_1, a_2, \cdots, a_d)$. By denoting $Vw = \widetilde{w}$, $Vu = \widetilde{u}$, and $\widetilde{w}_i\widetilde{u}_i = b_i$, we have

$$w^T A u = (Vw)^T \Sigma (Vu) = \sum_{i=1}^{d} a_i \widetilde{w}_i \widetilde{u}_i = \sum_{i=1}^{d} a_i b_i.$$

Let us denote σ as the permutation of $\{1, 2, \cdots, d\}$ such that

$$b_{\sigma(1)} \leq b_{\sigma(2)} \leq \cdots \leq b_{\sigma(d)}.$$

Here, we notice that

$$\sum_{i=1}^{d} b_i = \sum_{i=1}^{d} \widetilde{w}_i \widetilde{u}_i = (\widetilde{w})^T \widetilde{u} = (Vw)^T (Vu) = w^T u = 0.$$

Thus, there is at least one positive integer k such that $b_{\sigma(k)} \leq 0$ and $b_{\sigma(k+1)} \geq 0$. That is,

$$\sum_{i=1}^{k} -b_{\sigma(i)} = \sum_{i>k}^{d} b_{\sigma(i)} = \frac{1}{2}\sum_{i=1}^{d} |b_i|.$$

By using the rearrangement inequality, we have

$$\begin{aligned}\sum_{i=1}^{d} a_i b_i &\geq \sum_{i=1}^{d} a_i b_{\sigma(i)} = \sum_{i=1}^{k}(-a_i)\left(-b_{\sigma(i)}\right) + \sum_{i>k}^{d} a_i b_{\sigma(i)} \\ &\geq \sum_{i=1}^{k}(-a_1)\left(-b_{\sigma(i)}\right) + \sum_{i>k}^{d} a_d b_{\sigma(i)} = (a_d - a_1)\sum_{i=1}^{k}\left(-b_{\sigma(i)}\right) \\ &= \frac{a_d - a_1}{2}\sum_{i=1}^{d}|b_i| = \frac{a_d - a_1}{2}\sum_{i=1}^{d}|\widetilde{w}_i\widetilde{u}_i| \\ &= \frac{a_d - a_1}{2}|\widetilde{w}|^T|\widetilde{u}| \geq \frac{a_d - a_1}{2}\|\widetilde{w}\|\|\widetilde{u}\| = \frac{a_d - a_1}{2}\|w\|\|u\|,\end{aligned}$$

where $|v| = (|v_1|, |v_2|, \cdots, |v_d|)$ for any $v \in \mathbb{R}^d$. □

Now, we have the following proof for Proposition 5.

Proof Given $\Sigma_X = cI_{d_x}$, we first denote that $A := \langle xx^T\rangle_N = \Sigma_X + (\langle xx^T\rangle_N - \Sigma_X) = cI_{d_x} + \|\Sigma_X - \langle xx^T\rangle_N\|B$ with $\|B\| = 1$. According to the convergence of correlated matrix [2,11], we have $0 \leq \epsilon := \|\Sigma_X - \langle xx^T\rangle_N\| \leq \frac{c}{2}$ with high probability if N is large enough. That is, we have

$$(c-\epsilon)\|u\|^2 \leq u^T A u \leq (c+\epsilon)\|u\|^2 \quad \text{and} \quad a_d - a_1 \geq -2\epsilon, \tag{A.3}$$

for any $u \in \mathbb{R}^{d_x}$ if N is large enough. Here, $a_1 \geq a_2 \geq \cdots \geq a_{d_x}$ denote the eigenvalues of A.

For simplicity, it is equivalent to prove that $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w}\cdot n_E \leq 0$ for any $\boldsymbol{w} = (w_x, w_y) \in E$ and $\|w_x - w_x^*\| \geq \frac{2\sqrt{3}}{c}\|w_x^*\|\epsilon$ for any $0 \leq \epsilon \leq \frac{c}{2}$, where n_E denotes the exterior normal direction of E at $\boldsymbol{w}$.

In addition, we recall that $\boldsymbol{w} \in E$ if and only if $w_x^T\left(A(w_x - w_x^*)\right) = 0$. Thus, for any $\boldsymbol{w} \in E$, we have

$$\left(\frac{\sqrt{c+\epsilon}}{2}\|w_x^*\|\right)^2 \geq \frac{(w_x^*)^T A w_x^*}{4} = \left(w_x - \frac{w_x^*}{2}\right)^T A\left(w_x - \frac{w_x^*}{2}\right) \geq \left(\sqrt{c-\epsilon}\|w_x - \frac{w_x^*}{2}\|\right)^2,$$

which leads to

$$\frac{\sqrt{c+\epsilon}}{2}\|w_x^*\| \geq \sqrt{c-\epsilon}\|w_x - \frac{w_x^*}{2}\| \geq \sqrt{c-\epsilon}\left(\|w_x\| - \frac{\|w_x^*\|}{2}\right).$$

That is, we have

$$\|w_x\| \leq \frac{\sqrt{c+\epsilon} + \sqrt{c-\epsilon}}{2\sqrt{c-\epsilon}}\|w_x^*\| \leq \frac{\sqrt{c+\epsilon}}{\sqrt{c-\epsilon}}\|w_x^*\| \tag{A.4}$$

for any $\boldsymbol{w} \in E$.

Now, let us check the sign of $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w}(t)\cdot n_E$ if $\boldsymbol{w}(t) \in E$ while $\|w_x(t) - w_x^*\| \geq \frac{2\sqrt{3}}{c}\|w_x^*\|\epsilon$. First, we notice that

$$n_E = \nabla_{\boldsymbol{w}}\left(w_x^T\frac{\partial J^e}{\partial w_x}\right) = \begin{pmatrix} A(2w_x - w_x^*) \\ 0 \end{pmatrix}.$$

Thus, we have

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{w}\cdot n_E &= \left(A(2w_x - w_x^*)\right)^T\frac{\mathrm{d}}{\mathrm{d}t}w_x \\ &= -\|\boldsymbol{w}\|^{2-\frac{2}{L}}\left(A(2w_x - w_x^*)^T A(w_x - w_x^*)\right) \\ &= -\|\boldsymbol{w}\|^{2-\frac{2}{L}}\left((A(w_x - w_x^*))^T A(w_x - w_x^*) + w_x^T A^2(w_x - w_x^*)\right) \\ &= -\|\boldsymbol{w}\|^{2-\frac{2}{L}}\left(\|A(w_x - w_x^*)\|^2 + w_x^T A^2(w_x - w_x^*)\right).\end{aligned}$$

for any $\boldsymbol{w} \in E$. Given Lemma 5 and (A.3), we have

$$w_x^T A^2(w_x - w_x^*) = w_x^T A\left(A(w_x - w_x^*)\right) \geq -\epsilon\|w_x\|\|A(w_x - w_x^*)\|, \tag{A.5}$$

since $w_x^T\left(A(w_x - w_x^*)\right) = 0$. In the end, by combining the Lemma 5, (A.3), (A.4), and (A.5), we have

$$\begin{aligned}
&\|A(w_x - w_x^*)\|^2 + w_x^T A^2(w_x - w_x^*) \\
&\quad \geq \|A(w_x - w_x^*)\|\left(\|A(w_x - w_x^*)\| - \epsilon\|w_x\|\right) \\
&\quad \geq \|A(w_x - w_x^*)\|\left((c-\epsilon)\|w_x - w_x^*\| - \epsilon\frac{\sqrt{c+\epsilon}}{\sqrt{c-\epsilon}}\|w_x^*\|\right) \\
&\quad \geq \|A(w_x - w_x^*)\|\left(\frac{c}{2}\|w_x - w_x^*\| - \epsilon\sqrt{3}\|w_x^*\|\right) \geq 0
\end{aligned}$$

since $0 \leq \epsilon \leq \frac{c}{2}$ and $\|w_x - w_x^*\| \geq \frac{2\sqrt{3}}{c}\|w_x^*\|\epsilon$. This finishes the proof. □

A.6 Proof of Lemma 1

Proof First, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2 = -2\|\boldsymbol{w}\|^{-\frac{2}{L}}\left(\|\boldsymbol{w}\|^2 + (L-1)\|w_x\|^2\right)\left(w_x^T\frac{\partial J^e}{\partial w_x}\right).$$

This shows that $\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2$ has the opposite sign to $w_x^T\frac{\partial J^e}{\partial w_x}$. Because of Assumption 2 and the continuity of $\boldsymbol{w}(t)$, we see that $w_x^T\frac{\partial J^e}{\partial w_x}$ keeps the same sign to the initialization since $w_x^T\frac{\partial J^e}{\partial w_x} = 0$ if and only if $\boldsymbol{w} \in E$.

Similar proof for $\frac{\mathrm{d}}{\mathrm{d}t}\frac{\|w_x(t)\|^2}{\|w_y(t)\|^2}$ can be shown by calculating directly.

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\frac{\|w_x(t)\|^2}{\|w_y(t)\|^2} &= \frac{1}{\|w_y(t)\|^4}\left(\left(\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2\right)\|w_y(t)\|^2 - \left(\frac{\mathrm{d}}{\mathrm{d}t}\|w_y(t)\|^2\right)\|w_x(t)\|^2\right) \\
&= \frac{1}{\|w_y(t)\|^4}\left(\left(w_x^T(t)\frac{\mathrm{d}}{\mathrm{d}t}w_x(t)\right)\|w_y(t)\|^2 - \left(w_y^T(t)\frac{\mathrm{d}}{\mathrm{d}t}w_y(t)\right)\|w_x(t)\|^2\right) \\
&= -\|\boldsymbol{w}\|^{-\frac{2}{L}}\left(\frac{\|w_x(t)\|^2}{\|w_y(t)\|^2} + 1\right)\left(w_x^T\frac{\partial J^e(w)}{\partial w_x}\right) < 0.
\end{aligned}$$

Thus, $\frac{\mathrm{d}}{\mathrm{d}t}\frac{\|w_x(t)\|^2}{\|w_y(t)\|^2}$ also has the opposite sign to $w_x^T\frac{\partial J^e}{\partial w_x}$. □

A.7 Proof of Theorem 4

Proof Let first consider $\boldsymbol{w}(0) \in U^+$. Then, we have

$$\begin{aligned}
\|w_y(T)\|^2 - \|w_y(0)\|^2 &= \int_0^T \frac{\mathrm{d}}{\mathrm{d}t}\|w_y(t)\|^2\mathrm{d}t \\
&= \int_0^T \left(\frac{\mathrm{d}}{\mathrm{d}t}\|w_y(t)\|^2\mathrm{d}t / \frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2\right)\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2\mathrm{d}t \\
&= \int_0^T \frac{(L-1)\|w_y\|^2}{L\|w_x\|^2 + \|w_y\|^2}\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2\mathrm{d}t \quad (\boldsymbol{w}(t) \cap E = \emptyset) \\
&= \int_0^T \frac{L-1}{L(\|w_x\|^2/\|w_y\|^2) + 1}\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2\mathrm{d}t \quad (w_y(t) \neq 0) \\
&\leq \frac{(L-1)\|w_y(0)\|^2}{L\|w_x(0)\|^2 + \|w_y(0)\|^2}\int_0^T \frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2\mathrm{d}t \\
&= \frac{(L-1)\|w_y(0)\|^2}{L\|w_x(0)\|^2 + \|w_y(0)\|^2}\left(\|w_x(T)\|^2 - \|w_x(0)\|^2\right).
\end{aligned}$$

The inequality holds since $\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2 \le 0$ and $\frac{\mathrm{d}}{\mathrm{d}t}\frac{\|w_x\|^2}{\|w_y^2\|} \le 0$ for $0 \le t \le T$ if $\boldsymbol{w}(0) \in U^+$ according to Lemma 1. In addition, $\|w_y(T)\|^2 - \|w_y(0)\|^2 \le 0$ comes from the fact that $\|w_x(T)\|^2 \le \|w_x(0)\|^2$ since $\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2 \le 0$.

If $\boldsymbol{w}(0) \in E^-$, we have $\frac{\mathrm{d}}{\mathrm{d}t}\|w_x(t)\|^2 \ge 0$ and $\frac{\mathrm{d}}{\mathrm{d}t}\frac{\|w_x\|^2}{\|w_y^2\|} \ge 0$ for $0 \le t \le T$ according to Lemma 1. Thus, we can prove it with the same calculation above. □

B Proofs for ReLU DNNs

B.1 Proof of Lemma 3

Proof For any i and fixed $y \in \mathbb{R}^{d_y}$, let us first assume $W^1_{i,y}y \ge 0$. Given the definition of ReLU activation function, we may consider the following four sets:

$$\{\overline{W}^1_{i,x}x + \overline{b}^1_i \ge 0\} \cap \{\overline{W}^1_{i,x}x + \overline{b}^1_i + W^1_{i,y}y \ge 0\},$$
$$\{\overline{W}^1_{i,x}x + \overline{b}^1_i \ge 0\} \cap \{\overline{W}^1_{i,x}x + \overline{b}^1_i + W^1_{i,y}y \le 0\},$$
$$\{\overline{W}^1_{i,x}x + \overline{b}^1_i \le 0\} \cap \{\overline{W}^1_{i,x}x + \overline{b}^1_i + W^1_{i,y}y \ge 0\},$$
$$\{\overline{W}^1_{i,x}x + \overline{b}^1_i \le 0\} \cap \{\overline{W}^1_{i,x}x + \overline{b}^1_i + W^1_{i,y}y \le 0\},$$

to calculate $e_i(x, y)$ explicitly. Since $W^1_{i,y}y \ge 0$, we have

$$\{\overline{W}^1_{i,x}x + \overline{b}^1_i + W^1_{i,y}y \ge 0\} \subset \{\overline{W}^1_{i,x}x + \overline{b}^1_i \ge 0\},$$

which means

$$\{\overline{W}^1_{i,x}x + \overline{b}^1_i \ge 0\} \cap \{\overline{W}^1_{i,x}x + \overline{b}^1_i + W^1_{i,y}y \le 0\} = \emptyset.$$

In addition, we have $e_i(x, y) = 0$ on $\{\overline{W}^1_{i,x}x + \overline{b}^1_i \le 0\} \cap \{\overline{W}^1_{i,x}x + \overline{b}^1_i + W^1_{i,y}y \le 0\}$. As a result, we focus only on

$$\Omega^+_{i,x} := \{\overline{W}^1_{i,x}x + \overline{b}^1_i \ge 0\} \cap \Omega_x \tag{B.1}$$
$$\Omega^-_{i,x} := \{\overline{W}^1_{i,x}x + \overline{b}^1_i \le 0\} \cap \{\overline{W}^1_{i,x}x + \overline{b}^1_i + W^1_{i,y}y \ge 0\} \cap \Omega_x. \tag{B.2}$$

Then, if follows that

$$\begin{aligned}\|e_i(x, y)\|^2_{L^2(\Omega_x)} &= \int_{\Omega_x} |e_i(x, y)|^2 \mathrm{d}x \\ &= \underbrace{\int_{\Omega^+_{i,x}} |\overline{W}^2_i W^1_{i,y}y|^2 \mathrm{d}x}_{I^+} + \underbrace{\int_{\Omega^-_{i,x}} |\overline{W}^2_i(\overline{W}^1_{i,x}x + \overline{b}^1_i + W^1_{i,y}y)|^2 \mathrm{d}x}_{I^-}.\end{aligned}$$

For I^+, we have

$$\int_{\Omega^+_{i,x}} |\overline{W}^2_i W^1_{i,y}y|^2 \mathrm{d}x = |W^1_{i,y}y|^2 \int_{\Omega^+_{i,x}} |\overline{W}^2_i|^2 \mathrm{d}x = \frac{|W^1_{i,y}y|^2 \|\nabla h_i(x)\|^2_{L^2(\Omega_x)}}{\|\overline{W}^1_{i,x}\|^2}$$

because of the definition of $h_i(x)$ and the truncation property of ReLU activation function.

For I^-, we first denote a series of parallel hyperplanes in $\mathbb{R}^{d_x}$ as

$$H_{i,s} = \{x \mid \overline{W}^1_{i,x}x + \overline{b}^1_i + sW^1_{i,y}y = 0\}.$$

Then, we define

$$\mathcal{P}_{H_{i,s}}(\Omega_x) := \{\mathcal{P}_{H_{i,s}}(x) | x \in \Omega_x\} \subset H_{i,s}$$

as the projection of Ω_x onto $H_{i,s}$. We notice that $\mathcal{P}_{H_{i,s}}(\Omega_x)$ have the same measure for all $s \in [0,1]$. Finally, we define $\widetilde{\Omega}_{i,x}^-$ as the right cylinder which uses $\mathcal{P}_{H_{i,0}}(\Omega_x)$ and $\mathcal{P}_{H_{i,1}}(\Omega_x)$ as its bases. More precisely, we can write it as

$$\widetilde{\Omega}_{i,x}^- := \left\{ x \in \mathcal{P}_{H_{i,s}}(\Omega_x) \mid s \in [0,1] \right\}. \tag{B.3}$$

Here, we present Fig. 13 as a diagram about $\widetilde{\Omega}_{i,x}^-$.

Then, we have the following estimate by the parameterization of $\widetilde{\Omega}_{i,x}^-$ in (B.3) and integral by substitution

$$\begin{aligned} I^- &\le \int_{\widetilde{\Omega}_{i,x}^-} |\overline{W}_i^2(\overline{W}_{i,x}^1 x + \overline{b}_i^1 + W_{i,y}^1 y)|^2 \mathrm{d}x \\ &= \int_0^1 \int_{\mathcal{P}_{H_{i,s}}(\Omega_x)} |\overline{W}_i^2(\overline{W}_{i,x}^1 x + \overline{b}_i^1 + W_{i,y}^1 y)|^2 \mathrm{d}\tilde{x} \frac{|W_{i,y}^1 y|}{\|\overline{W}_{i,x}^1\|} \mathrm{d}s \\ &= \int_0^1 \frac{|W_{i,y}^1 y|}{\|\overline{W}_{i,x}^1\|} |\overline{W}_i^2|^2 (1-s)^2 |W_{i,y}^1 y|^2 \left|\mathcal{P}_{H_{i,0}}(\Omega_x)\right| \mathrm{d}s \\ &= \left|\mathcal{P}_{H_{i,0}}(\Omega_x)\right| \frac{|\overline{W}_i^2|^2 |W_{i,y}^1 y|^3}{3\|\overline{W}_{i,x}^1\|} \le C_{d_x} \frac{|\overline{W}_i^2|^2 |W_{i,y}^1 y|^3}{3\|\overline{W}_{i,x}^1\|}. \end{aligned}$$

Here, we notice that $\mathrm{d}x = \mathrm{d}\tilde{x} \frac{|W_{i,y}^1 y|}{\|\overline{W}_{i,x}^1\|} \mathrm{d}s$ since the distance between $H_{i,0}$ and $H_{i,1}$ is $\frac{|W_{i,y}^1 y|}{\|\overline{W}_{i,x}^1\|}$. In addition, $\left|\mathcal{P}_{H_{i,0}}(\Omega_x)\right|$ denotes the measure of $\mathcal{P}_{H_{i,0}}(\Omega_x)$ and we have

$$C_{d_x} := \sup_H \left|\mathcal{P}_H(\Omega_x)\right| \ge \left|\mathcal{P}_{H_{i,0}}(\Omega_x)\right|$$

for any $i = 1:n$, where H means a hyperplane in $\mathbb{R}^{d_x}$.

If $W_{i,y}^1 y \le 0$, we denote

$$\Omega_{i,x}^+ := \{\overline{W}_{i,x}^1 x + \overline{b}_i^1 + W_{i,y}^1 y \ge 0\} \cap \Omega_x \tag{B.4}$$

$$\Omega_{i,x}^- := \{\overline{W}_{i,x}^1 x + \overline{b}_i^1 \ge 0\} \cap \{\overline{W}_{i,x}^1 x + \overline{b}_i^1 + W_{i,y}^1 y \le 0\} \cap \Omega_x. \tag{B.5}$$

Then, we still have I^+ and I^- correspondingly. For I^-, we can follow the same strategy by defining $\widetilde{\Omega}_{i,x}^-$ and calculate the integral by decomposition and substitution. For I^+, we notice that

$$\Omega_{i,x}^+ \subset \{\overline{W}_{i,x}^1 x + \overline{b}_i^1 \ge 0\} \cap \Omega_x$$

since $W_{i,y}^1 y \le 0$. This means

$$I^+ = \int_{\Omega_{i,x}^+} |\overline{W}_i^2 W_{i,y}^1 y|^2 \mathrm{d}x \le \int_{\{\overline{W}_{i,x}^1 x + \overline{b}_i^1 \ge 0\} \cap \Omega_x} |\overline{W}_i^2 W_{i,y}^1 y|^2 \mathrm{d}x.$$

Then, we have the same estimate results as for the case $W_{i,y}^1 y \ge 0$. This finishes the proof. □

B.2 Proof of Corollary 2

Proof Given the definition of $D(y)$ and the estimate in (3.6), we have

$$\left\|\triangle_y f(\cdot, y)\right\|_{L^2(\Omega_x)}^2 \le \left(\prod_{\ell=3}^{L} \left\|\overline{W}^\ell\right\|^2\right) \sum_{\substack{i=1:n_2 \\ j=1:n_1}} \sum_{k=1:d_y} \left(\left|\left[W_{j,y}^1\right]_k\right|^2 + \left|\left[W_{j,y}^1\right]_k\right|^3 \right) D(y).$$

Since $\left[W^1_{j,y}\right]_k \sim \mathcal{N}(0, v^2)$ for all $j = 1 : n_1$ and $k = 1 : d_y$, it follows by the Monte Carlo estimate that with high probability there exists $\widetilde{D}_1$ such that

$$\frac{1}{n_2 n_2 d_y} \sum_{\substack{i=1:n_2 \\ j=1:n_1 \\ k=1:d_y}} \left|\left[W^1_{j,y}\right]_k\right|^2 - \mathbb{E}\left[\left|\left[W^1_{j,y}\right]_k\right|^2\right] \leq \widetilde{D}_1 \frac{1}{\sqrt{n_2 n_1 d_y}}.$$

This leads to

$$\sum_{\substack{i=1:n_2 \\ j=1:n_1 \\ k=1:d_y}} \left|\left[W^1_{j,y}\right]_k\right|^2 \leq n_2 n_1 d_y \mathbb{E}\left[\left|\left[W^1_{j,y}\right]_k\right|^2\right] + \widetilde{D}_1 \sqrt{n_2 n_1 d_y}$$

$$= n_2 n_1 d_y v^2 + \widetilde{D}_1 \sqrt{n_2 n_1 d_y},$$

with high probability. Similarly, we have

$$\sum_{\substack{i=1:n_2 \\ j=1:n_1 \\ k=1:d_y}} \left|\left[W^1_{j,y}\right]_k\right|^3 \leq n_2 n_1 d_y 2\sqrt{\frac{2}{\pi}} v^3 + \widetilde{D}_2 \sqrt{n_2 n_1 d_y}.$$

This proof is completed by taking $\widetilde{D} = \widetilde{D}_1 + \widetilde{D}_2$. □

Received: 8 April 2022 Accepted: 21 January 2023
Published online: 21 February 2023

References

1. Abdi, H., Williams, L.J.: Principal component analysis. Wiley Interdisciplinary Rev.: Comput. Stat. **2**(4), 433–459 (2010)
2. Adamczak, R., Litvak, A., Pajor, A., Tomczak-Jaegermann, N.: Quantitative estimates of the convergence of the empirical covariance matrix in log-concave ensembles. J. Am. Math. Soc. **23**(2), 535–561 (2010)
3. Arora, S., Cohen, N., Hazan, E.: On the optimization of deep networks: implicit acceleration by overparameterization. In: International Conference on Machine Learning, pp. 244–253
4. Arora, S., Cohen, N., Hu, W., Luo, Y.: Implicit regularization in deep matrix factorization. Adv. Neural. Inf. Process. Syst. **32**, 7413–7424 (2019)
5. Bah, B., Rauhut, H., Terstiege, U., Westdickenberg, M.: Learning deep linear neural networks: Riemannian gradient flows and convergence to global minimizers. Inf. Inference: J. IMA (2021). https://doi.org/10.1093/imaiai/iaaa039
6. Balakrishnama, S., Ganapathiraju, A.: Linear discriminant analysis-a brief tutorial. Inst. Signal Inf. Process. **18**(1998), 1–8 (1998)
7. Barron, A.R.: Universal approximation bounds for superpositions of a sigmoidal function. IEEE Trans. Inf. Theory **39**(3), 930–945 (1993)
8. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003)
9. Bishop, C.M., Nasrabadi, N.M.: Pattern Recognition and Machine Learning, vol. 4. Springer, Berlin (2006)
10. Bourgain, J., Dilworth, S., Ford, K., Konyagin, S., Kutzarova, D.: Explicit constructions of rip matrices and related problems. Duke Math. J. **159**(1), 145–185 (2011)
11. Cai, T.T., Zhang, C.H., Zhou, H.H.: Optimal rates of convergence for covariance matrix estimation. Ann. Stat. **38**(4), 2118–2144 (2010)
12. Chen, M., Jiang, H., Liao, W., Zhao, T.: Efficient approximation of deep relu networks for functions on low dimensional manifolds. Adv. Neural Inf. Process. Syst. **32** (2019)
13. Chen, Q., Hao, W., He, J.: A weight initialization based on the linear product structure for neural networks. Appl. Math. Comput. **415**, 126722 (2022)
14. Chen, Y., Dong, B., Xu, J.: Meta-mgnet: meta multigrid networks for solving parameterized partial differential equations. arXiv preprint arXiv:2010.14088 (2020)
15. Chu, J., Tsai, R.: Volumetric variational principles for a class of partial differential equations defined on surfaces and curves. Res. Math. Sci. **5**(2), 1–38 (2018)
16. Chui, C.K., Mhaskar, H.N.: Deep nets for local manifold learning. Front. Appl. Math. Stat. **4**, 12 (2018)
17. Cloninger, A., Klock, T.: A deep network construction that adapts to intrinsic dimensionality beyond the domain. Neural Netw. **141**, 404–419 (2021)
18. Cox, M.A.A., Cox, T.F.: Multidimensional scaling. In: Handbook of Data Visualization. Springer, Berlin, pp. 315–347 (2008)
19. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Process. Mag. **29**(6), 141–142 (2012)
20. Donoho, D.L., Grimes, C.: Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. Proc. Natl. Acad. Sci. **100**(10), 5591–5596 (2003)
21. Fefferman, C., Mitter, S., Narayanan, H.: Testing the manifold hypothesis. J. Am. Math. Soc. **29**(4), 983–1049 (2016)

22. Fukumizu, K.: Dynamics of batch learning in multilayer neural networks. In: International Conference on Artificial Neural Networks. Springer, Berlin, pp. 189–194 (1998)
23. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010)
24. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
25. Hacohen, G., Weinshall, D.: Principal components bias in deep neural networks. arXiv preprint arXiv:2105.05553 (2021)
26. Hardoon, D.R., Szedmak, S., Shawe-Taylor, J.: Canonical correlation analysis: an overview with application to learning methods. Neural Comput. **16**(12), 2639–2664 (2004)
27. He, J., Xu, J.: Mgnet: a unified framework of multigrid and convolutional neural network. Sci. China Math. **62**(7), 1331–1354 (2019)
28. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
29. Hein, M., Maier, M.: Manifold denoising. Advances in neural information processing systems **19** (2006)
30. Hsieh, J.T., Zhao, S., Eismann, S., Mirabella, L., Ermon, S.: Learning neural pde solvers with convergence guarantees. In: International Conference on Learning Representations (2019)
31. Johnson, W.B., Lindenstrauss, J.: Extensions of lipschitz mappings into a hilbert space 26. Contemporary Math. **26** (1984)
32. Kawaguchi, K.: Deep learning without poor local minima. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 586–594 (2016)
33. Kohn, K., Merkh, T., Montúfar, G., Trager, M.: Geometry of linear convolutional networks. arXiv preprint arXiv:2108.01538 (2021)
34. Krahmer, F., Ward, R.: New and improved Johnson-Lindenstrauss embeddings via the restricted isometry property. SIAM J. Math. Anal. **43**(3), 1269–1281 (2011)
35. Kublik, C., Tanushev, N.M., Tsai, R.: An implicit interface boundary integral method for poisson's equation on arbitrary domains. J. Comput. Phys. **247**, 279–311 (2013)
36. Liu, H., Chen, M., Zhao, T., Liao, W.: Besov function approximation and binary classification on low-dimensional manifolds using convolutional residual networks. In: International Conference on Machine Learning, pp. 6770–6780 (2021)
37. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
38. Ma, C., Wu, L., Weinan, E.: The slow deterioration of the generalization error of the random feature model. In: Mathematical and Scientific Machine Learning, pp. 373–389 (2020)
39. Nguegnang, G.M., Rauhut, H., Terstiege, U.: Convergence of gradient descent for learning linear neural networks. arXiv e-prints pp. arXiv-2108 (2021)
40. Narayanan, H., Mitter, S.: Sample complexity of testing the manifold hypothesis. In: Proceedings of the 23rd International Conference on Neural Information Processing Systems-Volume 2, pp. 1786–1794 (2010)
41. Nguyen, H., Tsai, R.: Numerical wave propagation aided by deep learning. J. Comput. Phys. **475**, 111828 (2023)
42. Niyogi, P., Smale, S., Weinberger, S.: Finding the homology of submanifolds with high confidence from random samples. Discrete Comput. Geom. **39**(1), 419–441 (2008)
43. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: an imperative style, high-performance deep learning library. Adv. Neural. Inf. Process. Syst. **32**, 8026–8037 (2019)
44. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
45. Saul, L.K., Roweis, S.T.: Think globally, fit locally: unsupervised learning of low dimensional manifolds. J. Mach. Learn. Res. **4**, 119–155 (2003)
46. Saxe, A.M., McClelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv preprint arXiv:1312.6120 (2013)
47. Schmidt-Hieber, J.: Deep relu network approximation of functions on a manifold. arXiv preprint arXiv:1908.00695 (2019)
48. Shaham, U., Cloninger, A., Coifman, R.R.: Provable approximation properties for deep neural networks. Appl. Comput. Harmon. Anal. **44**(3), 537–557 (2018)
49. Shen, Z., Yang, H., Zhang, S.: Optimal approximation rate of relu networks in terms of width and depth. J. de Math. Pures et Appl. **157**, 101–135 (2022)
50. Siegel, J.W., Xu, J.: Sharp bounds on the approximation rates, metric entropy, and n-widths of shallow neural networks (2021)
51. Steinerberger, S.: Randomized kaczmarz converges along small singular vectors. SIAM J. Matrix Anal. Appl. **42**(2), 608–615 (2021)
52. Stewart, G.W.: Matrix perturbation theory (1990)
53. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
54. Tretter, C.: Spectral Theory of Block Operator Matrices and Applications. World Scientific, Singapore (2008)
55. Weinberger, K.Q., Sha, F., Saul, L.K.: Learning a kernel matrix for nonlinear dimensionality reduction. In: Proceedings of the Twenty-first International Conference on Machine Learning, p. 106 (2004)
56. Yao, Y., Rosasco, L., Caponnetto, A.: On early stopping in gradient descent learning. Constr. Approx. **26**(2), 289–315 (2007)
57. Yarotsky, D.: Error bounds for approximations with deep relu networks. Neural Netw. **94**, 103–114 (2017)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.