Training Linear Neural Networks: Non-Local Convergence and Complexity Results

Armin Eftekhari [1]

Abstract

Linear networks provide valuable insights into the workings of neural networks in general. This paper identifies conditions under which the gradient flow provably trains a linear network, in spite of the non-strict saddle points present in the optimization landscape. This paper also provides the computational complexity of training linear networks with gradient flow. To achieve these results, this work develops a machinery to provably identify the stable set of gradient flow, which then enables us to improve over the state of the art in the literature of linear networks (Bah et al., 2019; Arora et al., 2018a). Crucially, our results appear to be the first to break away from the lazy training regime which has dominated the literature of neural networks. This work requires the network to have *a* layer with one neuron, which subsumes the networks with a scalar output, but extending the results of this theoretical work to all linear networks remains a challenging open problem.

1. Introduction and Overview

Consider the training samples and their labels $\{x_i, y_i\}_{i=1}^m \subset \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, respectively. By concatenating $\{x_i\}_i$ and $\{y_i\}_i$, we form the data matrices

$$X \in \mathbb{R}^{d_x \times m}, \qquad Y \in \mathbb{R}^{d_y \times m}. \tag{1}$$

Consider also a linear network, i.e., a neural network where the nonlinear activation functions are replaced with the identity map. To be specific, with N layers and the corresponding weight matrices $\{W_i\}_{i=1}^N$, this network is characterized by the linear map

$$\begin{aligned} &\mathbb{R}^{d_x} \to \mathbb{R}^{d_y} \\ &x \to Wx, \end{aligned} \tag{2}$$

and the matrix $W \in \mathbb{R}^{d_y \times d_x}$ in (2) is itself specified with the nonlinear (and often over-parametrized) map

$$\begin{aligned} &\mathbb{R}^{d_1 \times d_0} \times \cdots \times \mathbb{R}^{d_N \times d_{N-1}} \longrightarrow \mathbb{R}^{d_y \times d_x} \\ &(W_1, \cdots, W_N) \longrightarrow W := W_N \cdots W_1, \end{aligned} \tag{3}$$

where we set $d_0 = d_x$ and $d_N = d_y$ for consistency.

In foregoing the full generality of nonlinear neural networks, linear networks afford us a level of insight and technical rigor that is out of the reach for nonlinear networks, at least with our current technical tools (Arora et al., 2018b; Yan et al., 1994; Kawaguchi, 2016; Chitour et al., 2018; Trager et al., 2019; Saxe et al., 2013; Lu & Kawaguchi, 2017; Yun et al., 2017).

Indeed, despite the absence of activation functions, matrix W in the linear network (2,3) is a nonlinear function of $\{W_i\}_i$, and training this linear network thus involves solving a nonconvex optimization problem in $\{W_i\}_i$, which shares many interesting features of the nonlinear neural networks. Simply put, we cannot claim to understand neural networks in general without understanding linear networks.

Training the linear network (2,3) with the data (X, Y) in (1) can be cast as the optimization problem

$$\begin{cases} \min_{W_1,\cdots,W_N} \frac{1}{2}\|Y - W_N W_{N-1} \cdots W_1 X\|_F^2 \\ \text{subject to } W_j \in \mathbb{R}^{d_j \times d_{j-1}} \qquad \forall j \in [N], \end{cases} \tag{4}$$

which is nonconvex when $N \geq 2$. Above, $[N] = \{1, \cdots, N\}$. Let us introduce the shorthand

$$\begin{aligned} W_{\mathbf{N}} &:= (W_1, \cdots, W_N) \\ &\in \mathbb{R}^{d_1 \times d_0} \times \cdots \times \mathbb{R}^{d_N \times d_{N-1}} =: \mathbb{R}^{d_{\mathbf{N}}}, \end{aligned} \tag{5}$$

which allows us to rewrite problem (4) more compactly as

$$\min_{W_{\mathbf{N}}} L_N(W_{\mathbf{N}}) \text{ subject to } W_{\mathbf{N}} \in \mathbb{R}^{d_{\mathbf{N}}}, \tag{6}$$

where $L_N(W_{\mathbf{N}}) := \frac{1}{2}\|Y - W_N \cdots W_1 X\|_F^2$. With this setup and before turning to the details, let us highlight the contributions of this paper, in the order of appearance.

---

[1]Department of Mathematics and Mathematical Statistics, Umea University, Sweden. AE is indebted to Holger Rauhut, Ulrich Terstiege and Gongguo Tang for insightful discussions. Correspondence to: Armin Eftekhari <armin.eftekhari@umu.se>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

- Theorem 2.8 in Section 2 provides a new analysis of the optimization landscape of linear networks, where we uncover a previously-unknown link to the celebrated Eckart-Young-Mirsky theorem and the geometry of the principal component analysis (PCA).

- Theorem 3.8 in Section 3 identifies the conditions under which gradient flow successfully trains a linear network, despite the presence of non-strict saddle points in the optimization landscape.

  Theorem 3.8 thus improves the state of the art in (Bah et al., 2019) as the first convergence result outside of the lazy training regime, reviewed later. This improvement is achieved with a new argument that provably identifies the stable set of gradient flow, in the language of dynamical systems theory.

  Theorem 3.8 applies to linear networks that have *a* layer with a single neuron, see Assumption 3.6. This case corresponds to the popular spiked covariance model in statistics, and subsumes networks with a scalar output. Extension of Theorem 3.8 to all linear networks remains a challenging open problem, as there is no natural notion of stable set in general. We will however speculate about how Theorem 3.8 might serve as the natural building block for a more general result in the future.

- Theorem 4.4 in Section 4 quantifies the computational complexity of training a linear network by establishing *non-local* convergence rates for gradient flow. This result also quantifies for the first time how the (faraway) convergence rate benefits from increasing the network depth.

  Theorem 4.4 presents the first convergence rate for linear networks outside of the lazy training regime, thus improving the state of the art in (Arora et al., 2018a).

  Indeed, of the dozens of related works, virtually all belong to this lazy regime, to the best of our knowledge, thus signifying the importance of this breakthrough. Theorem 4.4 also corresponds to the spiked covariance model.

2. Landscape of Linear Networks

The landscape of nonconvex program (6) has been widely studied in the literature, with contributions from (Arora et al., 2018b;a; Chitour et al., 2018; Bartlett et al., 2019; Saxe et al., 2013; Hardt & Ma, 2016; Laurent, 2018; Trager et al., 2019; Baldi & Hornik, 1989; Zhu et al., 2019; He et al., 2016; Nguyen, 2019). The state of the art here is Proposition 31 in (Bah et al., 2019), reviewed in Appendix E, which itself improves over Theorem 2.3 in (Kawaguchi, 2016).

The main result of this section, Theorem 2.8 below, is a variation of Proposition 31 in (Bah et al., 2019) with an additional assumption. In Section 3, we will use Theorem 2.8 to improve the state of art for training linear networks, under this new assumption.

Crucially, the proof of Theorem 2.8 uncovers a previously-unknown link to the celebrated Eckart-Young-Mirsky theorem (Eckart & Young, 1936; Mirsky, 1966) and the geometry of PCA.

To begin, let us concretely define the notion of optimality for problem (6).

Definiton 2.1 (First-order stationarity for (6))**.** *We say that $\overline{W}_{\mathbf{N}} \in \mathbb{R}^{d_{\mathbf{N}}}$ is a first-order stationary point (FOSP) of problem (6) if*

$$\nabla L_N(\overline{W}_{\mathbf{N}}) = 0, \tag{7}$$

where $\nabla L_N(\overline{W}_{\mathbf{N}})$ is the gradient of L_N at $\overline{W}_{\mathbf{N}}$.

Definiton 2.2 (Second-order stationarity for (6))**.** *We say that $\overline{W}_{\mathbf{N}} \in \mathbb{R}^{d_{\mathbf{N}}}$ is a second-order stationary point (SOSP) of problem (6) if, in addition to (7), it holds that*

$$\nabla^2 L_N(\overline{W}_{\mathbf{N}})[\Delta_{\mathbf{N}}] \geq 0, \qquad \forall \Delta_{\mathbf{N}} \in \mathbb{R}^{d_{\mathbf{N}}}, \tag{8}$$

where $\nabla^2 L_N(\overline{W}_{\mathbf{N}})[\Delta_{\mathbf{N}}]$ is the second derivative of L_N at $W_{\mathbf{N}}$ along the direction $\Delta_{\mathbf{N}}$.

Definiton 2.3 (Strict saddles of (6))**.** *Any FOSP of problem (6), which is not an SOSP, is a strict saddle point of (6).*

Any SOSP of problem (6) is either a local minimizer of (6), or a non-strict saddle point of problem (6). Unlike a non-strict saddle point, there always exists a descent direction to escape from a strict saddle point (Lee et al., 2017). To continue, let

$$r := \min_{j \leq N} d_j, \qquad \text{(smallest width of the network)} \tag{9}$$

denote the smallest width of the linear network (2,3). As shown in Appendix A, we can reformulate problem (6) as

$$\min_{W_{\mathbf{N}}} L_N(W_{\mathbf{N}})$$
$$= \begin{cases} \min_{W} \frac{1}{2}\|Y - WX\|_F^2 =: L_1(W) \\ \text{subject to } \operatorname{rank}(W) \leq r \end{cases} \tag{10a}$$
$$= \begin{cases} \min_{P,Q} \frac{1}{2}\|Y - PQX\|_F^2 =: L_2(P,Q) \\ \text{subject to } P \in \mathbb{R}^{d_y \times r},\ Q \in \mathbb{R}^{r \times d_x}. \end{cases} \tag{10b}$$

In particular, the notion of optimality for problem (10b) is defined similar to Definitions 2.1-2.3. There is a correspondence between the stationary points of problems (6) and (10b), proved in Appendix B.

Lemma 2.4 (Pairwise correspondence between SOSPs)**.** *Any FOSP $\overline{W}_{\mathbf{N}} = (\overline{W}_1, \cdots, \overline{W}_N)$ of problem (6) corresponds to an FOSP $(\overline{P}, \overline{Q})$ of problem (10b), provided* *that $\overline{W} = \overline{W}_N \cdots \overline{W}_1$ is rank-r. Moreover, any SOSP $\overline{W}_{\mathbf{N}}$ of problem (6) corresponds to an SOSP $(\overline{P}, \overline{Q})$ of problem (10b), provided that rank$(\overline{W}) = r$.*

Let $\mathcal{P}_X := X^\dagger X$ and $\mathcal{P}_{X^\perp} := I_m - \mathcal{P}_X$ denote the orthogonal projections onto the row span of X and its orthogonal complement, respectively. Here, $X^\dagger$ is the pseudo-inverse of X and $I_m \in \mathbb{R}^{m\times m}$ is the identity matrix. Using the decomposition $Y = Y\mathcal{P}_X + Y\mathcal{P}_{X^\perp}$, we can in turn rewrite problem (10b) as

$$\begin{aligned}
&\min_{P,Q} \; L_2(P,Q) \\
&= \frac{1}{2}\|Y\mathcal{P}_{X^\perp}\|_F^2 + \begin{cases} \min_{P,Q,Q'} \; \frac{1}{2}\|Y\mathcal{P}_X - PQ'\|_F^2 \\ \text{subject to } Q' = QX \end{cases} \\
&\geq \frac{1}{2}\|Y\mathcal{P}_{X^\perp}\|_F^2 + \min_{P,Q'} \; \frac{1}{2}\|Y\mathcal{P}_X - PQ'\|_F^2. \qquad (11)
\end{aligned}$$

The relaxation above is tight, and there is a correspondence between the stationary points, proved in Appendix C.

Lemma 2.5 (Pairwise correspondence between SOSPs)**.** *Suppose that $XX^\top$ is invertible. Then it holds that*

$$\begin{aligned}
&-\frac{1}{2}\|Y\mathcal{P}_{X^\perp}\|_F^2 + \min_{P,Q} \; L_2(P,Q) \\
&= \begin{cases} \min_{P,Q'} \; \frac{1}{2}\|Y\mathcal{P}_X - PQ'\|_F^2 =: L_2'(P,Q') \\ \text{subject to } P \in \mathbb{R}^{d_y\times r}, \; Q' \in \mathbb{R}^{r\times m}. \end{cases} \qquad (12)
\end{aligned}$$

Any FOSP $(\overline{P}, \overline{Q})$ of problem (10b) corresponds to an FOSP $(\overline{P}, \overline{Q}')$ of problem (12). Moreover, any SOSP $(\overline{P}, \overline{Q})$ of problem (10b) corresponds to an SOSP $(\overline{P}, \overline{Q}')$ of problem (12).

Note that solving problem (12) involves finding a best rank-r approximation of $Y\mathcal{P}_X$ or, equivalently, finding r leading principal components of $Y\mathcal{P}_X$ (Murphy, 2012). By combining Lemmas 2.4 and 2.5, we immediately reach the following conclusion.

Lemma 2.6 (Pairwise correspondence between SOSPs)**.** *Suppose that $XX^\top$ is invertible. Then any FOSP $\overline{W}_{\mathbf{N}}$ of problem (6) corresponds to an FOSP $(\overline{P}, \overline{Q}')$ of problem (12), provided that $\overline{W} = \overline{W}_N \cdots \overline{W}_1$ is rank-r. Moreover, any SOSP $\overline{W}_{\mathbf{N}}$ of problem (6) corresponds to an SOSP $(\overline{P}, \overline{Q}')$ of (12), provided that rank$(\overline{W}) = r$.*

We next recall a variant of the celebrated EYM theorem (Eckart & Young, 1936; Mirsky, 1966; Hauser & Eftekhari, 2018; Hauser et al., 2018), which specifies the landscape of the PCA problem (12).

Theorem 2.7 (EYM theorem)**.** *Any SOSP $(\overline{P}, \overline{Q}')$ of the PCA problem (12) is also a global minimizer of problem (12).*

With Lemma 2.6 at hand, we invoke Theorem 2.7 to uncover the landscape of problem (6), see Appendix D for the proof.

Theorem 2.8 (Landscape of linear networks)**.** *Suppose that $XX^\top$ is invertible and that rank$(YX^\dagger X) \geq r$. Then any SOSP $\overline{W}_{\mathbf{N}} = (\overline{W}_1, \cdots, \overline{W}_N)$ of problem (6) is a global minimizer of problem (6), provided that $\overline{W}_N \cdots \overline{W}_1$ is a rank-r matrix.*

In words, Theorem 2.8 identifies certain SOSPs of problem (6) which are global minimizers of problem (6). A few important remarks are in order: ① The proof of Theorem 2.8 establishes a pairwise correspondence with the stationary points and the geometry of the PCA problem. This connection was previously unknown, to the best of our knowledge.

② Any rank-degenerate SOSP of problem (6) is excluded from Theorem 2.8, i.e., any SOSP $\overline{W}_{\mathbf{N}} = (\overline{W}_1, \cdots, \overline{W}_N)$ such that rank$(\overline{W}_N \cdots \overline{W}_1) < r$ is excluded from our result. For example, the zero matrix is a spurious SOSP of problem (6) when the network depth $N \geq 2$, as observed in (Bah et al., 2019; Kawaguchi, 2016; Trager et al., 2019). The landscape of problem (6) in general is therefore more complicated than the special case of $N = 2$, corresponding to the Eckart-Young-Mirsky theorem, see Theorem 2.7.

③ Theorem 2.8 is a variation of Proposition 31 in (Bah et al., 2019) with a new assumption on $YX^\dagger X$, which will be necessary shortly. Similar assumptions have been used in the context of PCA, for example in (Helmke & Shayman, 1995). ④ For completeness, we also prove Theorem 2.8 using Proposition 31 in (Bah et al., 2019) as the starting point, see Appendix E.

3. Convergence of Gradient Flow

In view of Theorem 2.8 above, even though nonconvex, the landscape of problem (6) has certain favourable properties. On the other hand, problem (6) fails to satisfy the strict saddle property that enables first-order algorithms to avoid saddle points (Lee et al., 2016; Ge et al., 2015). For example, the zero matrix is a non-strict saddle point of problem (6) when the network depth $N \geq 2$, as discussed earlier.

Against this mixed background, it is natural to ask if first-order algorithms can successfully train linear neural networks. This fundamental question has remained unanswered in the literature, to our knowledge. Indeed, outside the lazy training regime, reviewed in Section 4, it is not known if gradient flow can successfully solve problem (6).

In fact, the state of the art here, Theorem 35(a) in (Bah et al., 2019), guarantees the convergence of gradient flow to a minimizer of L_N, when *restricted* to one of few regions in $\mathbb{R}^{d_{\mathbf{N}}}$. Even though these regions are known in advance, their result cannot predict which region would contain the limit point of gradient flow, for a given initialization.

In other words, Theorem 35(a) in (Bah et al., 2019) does *not* guarantee the convergence of gradient flow to a global minimzer of problem (6), and gradient flow might indeed converge to a spurious SOSP of problem (6), such as the zero matrix. For completeness, Theorem 35(a) in (Bah et al., 2019) is reviewed in Appendix F.

In an important setting, this section answers the open question of convergence of gradient flow for training linear networks. This is achieved with a new argument, which enables us to provably identify the stable set of gradient flow, in the language of dynamical systems theory. Let us begin with the necessary preparations.

Consider gradient flow applied to program (6), specified as

$$\dot{W}_j(t) = \frac{\mathrm{d}W_j(t)}{\mathrm{d}t} = -\nabla_{W_j} L_N\left(W_{\mathbf{N}}(t)\right), \quad \forall j \in [N],\ \forall t \geq 0, \qquad \text{(gradient flow)} \tag{13}$$

and initialized at $W_{\mathbf{N},0} \in \mathbb{R}^{d_{\mathbf{N}}}$. Above, $\nabla_{W_j} L_N$ is the gradient of L_N with respect to W_j, the weight matrix for the j^{th} layer of the linear network, see (4,5,6).

A consequence of the Lojasiewicz' theorem is the following convergence result for the gradient flow. See Appendix G for the proof, similar to Theorem 11 in (Bah et al., 2019).

Lemma 3.1 (Convergence, uninformative)**.** *If $XX^\top$ is invertible, then gradient flow (13) converges. Moreover, the limit point is an SOSP $\overline{W}_{\mathbf{N}} \in \mathbb{R}^{d_{\mathbf{N}}}$ of problem (6), for almost every initialization $W_{\mathbf{N},0}$ with respect to the Lebesgue measure in $\mathbb{R}^{d_{\mathbf{N}}}$.*

To study this limit point $\overline{W}_{\mathbf{N}}$, we focus here on a common initialization technique for linear networks (Hardt & Ma, 2016; Bartlett et al., 2019; Arora et al., 2018a;b).

Definiton 3.2 (Balanced initialization)**.** *For gradient flow (13), we call $W_{\mathbf{N},0} = (W_{1,0}, \cdots, W_{N,0}) \in \mathbb{R}^{d_{\mathbf{N}}}$ a balanced initialization if*

$$W_{j+1,0}^\top W_{j+1,0} = W_{j,0} W_{j,0}^\top, \qquad \forall j \in [N-1]. \tag{14}$$

Claim 4 in (Arora et al., 2018a) underscores the necessity of a (nearly) balanced initialization for linear networks. More generally, (Sutskever et al., 2013) highlights the importance of initialization in deep neural networks. The main result of this section, Theorem 3.8 below, thus requires an (exactly) balanced initialization.

Assuming exact balanced-ness in (14) is sufficient here because the focus of this theoretical work is continuous-time optimization. More generally, approximate balanced-ness is necessary for discretized algorithms, such as gradient descent (Arora et al., 2018a). We avoid this additional layer of complexity here as it does not seem to add any key theoretical insights to this paper.

A useful observation is that, if the initialization is balanced, gradient flow remains balanced afterwards, see for example Lemma 2 in (Bah et al., 2019). More formally, gradient flow (13) satisfies

$$\begin{aligned} &W_{j+1,0}^\top W_{j+1,0} = W_{j,0} W_{j,0}^\top, \qquad \forall j \in [N-1] \\ &\Longrightarrow W_{j+1}(t)^\top W_{j+1}(t) = W_j(t) W_j(t)^\top, \end{aligned} \tag{15}$$

for every $j \in [N-1]$ and every $t \geq 0$. Above, the weight matrix $W_j(t)$ is the j^{th} component of $W_{\mathbf{N}}(t)$, see (5).

Alongside gradient flow (13), it is convenient to introduce another flow (Bah et al., 2019; Arora et al., 2018a), which dictates the evolution of the end-to-end product of the weight matrices of the linear network.

Concretely, for a matrix $W \in \mathbb{R}^{d_y \times d_x}$, consider the linear operator $\mathcal{A}_W$ specified as

$$\begin{aligned} \mathcal{A}_W : &\mathbb{R}^{d_y \times d_x} \to \mathbb{R}^{d_y \times d_x} \\ &\Delta \to \sum_{j=1}^{N} (WW^\top)^{\frac{N-j}{N}} \Delta (W^\top W)^{\frac{j-1}{N}}. \end{aligned} \tag{16}$$

For a balanced initialization $W_{\mathbf{N},0} = (W_{1,0}, \cdots, W_{N,0})$, gradient flow (13) in $\mathbb{R}^{d_{\mathbf{N}}}$ induces a flow in $\mathbb{R}^{d_y \times d_x}$, initialized at $W_0 = W_{N,0} \cdots W_{1,0} \in \mathbb{R}^{d_y \times d_x}$ and specified as

$$\begin{aligned} \dot{W}(t) &= -\mathcal{A}_{W(t)}\left(\nabla L_1(W(t))\right) \qquad \forall t \geq 0, \\ &= -\mathcal{A}_{W(t)}(W(t)XX^\top - YX^\top) \qquad \text{(see (10a))} \\ &\qquad \text{(induced flow)} \end{aligned} \tag{17}$$

see for example Equation (26) in (Bah et al., 2019). Above,

$$W(t) = W_N(t) \cdots W_1(t) \in \mathbb{R}^{d_y \times d_x}. \tag{18}$$

We will refer to (17) as the *induced flow*, which governs the evolution of the end-to-end product of the weight matrices of the linear network.

It is known that induced flow (17) admits an analytic singular value decomposition (SVD), see for example Lemma 1 and Theorem 3 in (Arora et al., 2019a) or (Illashenko & Yakovenko, 2008). More specifically, it holds that

$$W(t) \overset{\text{SVD}}{=} \widetilde{U}(t)\widetilde{S}(t)\widetilde{V}(t)^\top, \qquad \forall t \geq 0, \tag{19}$$

provided that the network depth $N \geq 2$. In (19), $\widetilde{U}(t), \widetilde{V}(t), \widetilde{S}(t)$ are analytic functions of t (Parks & Krantz, 1992). Moreover, $\widetilde{U}(t) \in \mathbb{R}^{d_y \times d_y}, \widetilde{V}(t) \in \mathbb{R}^{d_x \times d_x}$ are orthonormal bases, and $\widetilde{S}(t) \in \mathbb{R}^{d_y \times d_x}$ contains the singular values of $W(t)$ in no specific order.

The evolution of the singular values of $W(t)$ in (19) is also known (Arora et al., 2019a; Townsend, 2016). In particular, the following byproduct about the rank of $W(t)$ is important for us, see Appendix H for the proof.

Lemma 3.3 (Rank-invariance)**.** *For induced flow* (17)*, rank$(W(t)) =$ rank(W_0) for all $t \geq 0$, provided that $XX^\top$ is invertible and the network depth $N \geq 2$.*

Let us henceforth assume that $XX^\top$ is invertible, and that gradient flow (13) is initialized at $W_{\mathbf{N},0} \in \mathcal{M}_{\mathbf{N},r}$, where

$$\mathcal{M}_{\mathbf{N},r} := \Big\{ W_{\mathbf{N}} : \operatorname{rank}(W_N \cdots W_1) = r \Big\} \subset \mathbb{R}^{d_{\mathbf{N}}}, \quad (20)$$

see (5). We now make the following observations about the set $\mathcal{M}_{\mathbf{N},r}$, proved in Appendix I.

Lemma 3.4 (Propeties of $\mathcal{M}_{\mathbf{N},r}$)**.** ① *$\mathcal{M}_{\mathbf{N},r}$ is not a closed subset of $\mathbb{R}^{d_{\mathbf{N}}}$.* ② *The complement of $\mathcal{M}_{\mathbf{N},r}$ in $\mathbb{R}^{d_{\mathbf{N}}}$ has Lebesgue measure zero. (In particular, $\mathcal{M}_{\mathbf{N},r}$ is a dense subset of $\mathbb{R}^{d_{\mathbf{N}}}$.)*

In view of Lemma 3.4, almost every initialization $W_{\mathbf{N},0} \in \mathbb{R}^{d_{\mathbf{N}}}$ of gradient flow (13) falls into the set $\mathcal{M}_{\mathbf{N},r}$, i.e.,

$$W_{\mathbf{N},0} \in \mathcal{M}_{\mathbf{N},r}, \qquad \text{almost surely.} \qquad (21)$$

Moreover, once initialized in $\mathcal{M}_{\mathbf{N},r}$ with a balanced initialization, induced flow (17) remains rank-r at all times by (18,20) and Lemma 3.3. Consequently, gradient flow (13) remains in $\mathcal{M}_{\mathbf{N},r}$ at all times, see again (18,20). We combine this last observation with (21) to conclude that

$$W_{\mathbf{N}}(t) \in \mathcal{M}_{\mathbf{N},r}, \qquad \forall t \geq 0, \qquad \text{almost surely,} \qquad (22)$$

over the choice of balanced initialization $W_{\mathbf{N},0} \in \mathbb{R}^{d_{\mathbf{N}}}$. Despite (22), the limit point $\overline{W}_{\mathbf{N}}$ of gradient flow (13) might *not* belong to $\mathcal{M}_{\mathbf{N},r}$ because $\mathcal{M}_{\mathbf{N},r}$ is *not* closed, see Lemma 3.4.

That is, even though the limit point $\overline{W}_{\mathbf{N}}$ of gradient flow is almost surely an SOSP of problem (6) by Lemma 3.1, we *cannot* apply Theorem 2.8 and $\overline{W}_{\mathbf{N}}$ might be an spurious SOSP of problem (6), such as the zero matrix. Indeed, Remark 39 in (Bah et al., 2019) constructs an example where $\overline{W}_{\mathbf{N}} \notin \mathcal{M}_{\mathbf{N},r}$, see also (Yan et al., 1994). To avoid this unwanted behaviour, it is necessary to restrict the initialization of the gradient flow and impose additional assumptions.

Our first assumption is that the data is statistically whitened, which is common in the analysis of linear networks (Arora et al., 2018a; Bartlett et al., 2019).

Definiton 3.5 (Whitened data)**.** *We say that the data matrix $X \in \mathbb{R}^{d_x \times m}$ is whitened if*

$$\frac{XX^\top}{m} = \frac{1}{m}\sum_{i=1}^{m} x_i x_i^\top = I_{d_x}, \qquad (23)$$

where $I_{d_x} \in \mathbb{R}^{d_x \times d_x}$ is the identity matrix.

Our second assumption is that $r = 1$ in (9). This case is significant as it corresponds to the popular *spiked covariance* model in statistics and signal processing (Eftekhari et al., 2019; Johnstone et al., 2001; Vershynin, 2012; Berthet et al., 2013; Deshpande & Montanari, 2014), to name a few.

Moreover, $r = 1$ subsumes the important case of networks with a scalar output.

Lastly, the case $r = 1$ appears to be the natural building block for the case $r > 1$ via a *deflation argument* (Mackey, 2009; Zhang & Ghaoui, 2011). Indeed, gradient flow (13) moves orthogonal to the principal directions that it has previously discovered or "peeled". Extending our results to the case $r > 1$ remains a challenging open problem.

From (10a) with $r = 1$, recall that problem (6) for training a linear neural network is closely related to the problem

$$\begin{aligned} &\min_W \frac{1}{2}\|Y\mathcal{P}_X - WX\|_F^2 \text{ subject to } \operatorname{rank}(W) \leq r = 1 \\ &= \min_W \frac{m}{2}\|Z - W\|_F^2 \text{ subject to } \operatorname{rank}(W) \leq 1, \end{aligned} \qquad (24)$$

where the second line above is obtained using (23), and

$$Z := \frac{YX^\top}{m}. \qquad (25)$$

We are in position to collect all the assumptions made in this section in one place.

Assumption 3.6. *In this section, we assume that the linear network* (2,3) *has depth $N \geq 2$, and one of the layers has only one neuron, i.e., $r = 1$ in* (9)*. Moreover, the data matrix X in* (1) *is whitened as in* (23)*, and $Z = \frac{1}{m}YX^\top$ in* (25) *satisfies*

$$\text{rank}(Z) \geq r = 1, \quad \gamma_Z := \frac{s_{Z,2}}{s_Z} < 1, \qquad (26)$$

where s_Z and $s_{Z,2}$ are the two largest singular values of Z. Lastly, we assume that the initialization of gradient flow (13) *is balanced, see Definition 3.2.*

In view of (26), let us define

$$Z_1 = u_Z \cdot s_Z \cdot v_Z^\top \qquad \text{(target matrix)} \qquad (27)$$

to be the best rank-1 approximation of Z, obtained via SVD. Here, $\|u_Z\|_2 = \|v_Z\|_2 = 1$, and s_Z appeared in (26).

Note that Z_1 is the unique solution of problem (24), because Z has a nontrivial spectral gap in (26), see for example Section 1 of (Golub et al., 1987).

Let us fix $\alpha \in [\gamma_Z, 1)$. To exclude the zero matrix as the limit point of gradient flow (13), the key is to restrict the initialization to a particular subset of the feasible set of problem (6) with $r = 1$, specified as

$$\mathcal{N}_{\mathbf{N},\alpha} := \Big\{ W_{\mathbf{N}} = (W_1, \cdots, W_N) : \\ W_N \cdots W_1 \overset{\text{tSVD}}{=} u_W \cdot s_W \cdot v_W^\top, \\ s_W > (\alpha - \gamma_Z) s_Z,\, u_W^\top Z_1 v_W > \alpha s_Z \Big\} \subset \mathbb{R}^{d_{\mathbf{N}}}, \quad (28)$$

where s_Z, γ_Z were defined in (26). Above, tSVD stands for the thin SVD. A simple observation is that the set $\mathcal{N}_{\mathbf{N},\alpha}$ has infinite Lebesgue measure in $\mathbb{R}^{d_{\mathbf{N}}}$.

Such restriction of the feasible set of problem (6) is *necessary* as described earlier, see also the negative example constructed in Remark 39 of (Bah et al., 2019). Crucially, note that the end-to-end matrices in $\mathcal{N}_{\mathbf{N},\alpha}$ are positively correlated with Z_1, and also bounded away from the origin.

An important observation is that, once initialized in $\mathcal{N}_{\mathbf{N},\alpha}$, gradient flow (13) avoids the zero matrix, see Appendix J.

Lemma 3.7 (Stable set)**.** *For gradient flow* (13) *initialized at $W_{\mathbf{N},0} \in \mathcal{N}_{\mathbf{N},\alpha}$, the limit point exists and satisfies $\overline{W}_{\mathbf{N}} \in \mathcal{M}_{\mathbf{N},1}$. Above, $\alpha \in (\gamma_Z, 1)$, and Assumption* 3.6 *and its notation are in force, see also* (20,28)*.*

Combining Lemma 3.7 with Lemma 3.1, we find that the limit point $\overline{W}_{\mathbf{N}} \in \mathcal{M}_{\mathbf{N},1}$ of gradient flow (13) is an SOSP of problem (6), for every balanced initialization $W_{\mathbf{N},0} \in \mathcal{N}_{\mathbf{N},\alpha}$ outside a subset with Lebesgue measure zero.

We finally invoke Theorem 2.8 to conclude that this SOSP $\overline{W}_{\mathbf{N}} \in \mathcal{M}_{\mathbf{N},1}$ is in fact a global minimizer of L_N in $\mathbb{R}^{d_{\mathbf{N}}}$. This conclusion is summarized below.

Theorem 3.8 (Convergence)**.** *Gradient flow* (13) *converges to a global minimizer of problem* (6) *from every balanced initialization in $\mathcal{N}_{\mathbf{N},\alpha} \subset \mathbb{R}^{d_{\mathbf{N}}}$, outside of a subset with Lebesgue measure zero, see* (28)*. Above, $\alpha \in (\gamma_Z, 1)$, and Assumption* 3.6 *and its notation are in force.*

A few important remarks are in order. ① Outside the lazy training regime reviewed in Section 4, to our knowledge, Theorem 3.8 is the first convergence result for linear networks, answering the fundamental question of when gradient flow successfully trains a linear network.

② Indeed, under Assumption 3.6, Theorem 3.8 improves over Theorem 35 in (Bah et al., 2019) which does *not* guarantee the convergence of gradient flow (13) to a solution of problem (6), and discussed earlier and also in Appendix F.

③ This improvement was achieved by provably restricting the gradient flow (13) to its stable set $\mathcal{N}_{\mathbf{N},\alpha}$ in (28), and such a restriction is indeed necessary as detailed earlier.

④ Note that Theorem 3.8 sheds light on the theoretical aspects of the training of neural networks, and should not be viewed as an initialization technique for linear networks. In turn, linear networks only serve to improve our theoretical understanding of neural networks in general.

Let us also examine the content of Assumption 3.6. ① The case $r = 1$ in (9) corresponds to the spiked covariance model in statistics, and covers the important case of networks with a scalar output. Lastly, $r = 1$ appears to be the natural building block for extension to $r > 1$, which remains an open problem, see the discussion after (23).

② The assumption of whitened data in (23) is commonly used in the context of linear networks, see for example (Arora et al., 2018a; Bartlett et al., 2019). ③ The requirement that $\text{rank}(Z) = \text{rank}(Y\mathcal{P}_X) \geq r = 1$ in Assumption 3.6 is evidently necessary to avoid the limit point of zero.

④ Finally, it is known that the induced flow (17) for an *unbalanced* initialization deviates rapidly from its balanced counterpart. It is therefore not clear if an unbalanced flow would provably avoid rank-degenerate limit points. However, we suspect that any disadvantage of an unbalanced initialization will disappear asymptotically as the network depth N grows larger, see Equation 8 in (Bah et al., 2019).

4. Convergence Rate of Gradient Flow

In view of Theorem 3.8, it is natural to ask how fast we can train a linear network with gradient flow. However, Theorem 3.8 is notably silent about the convergence *rate* of gradient flow (13) to a solution of problem (6). In short, is it possible for gradient flow to efficiently solve problem (6)?

As we review now, this fundamental question has not been answered in the literature beyond the lazy training regime. Indeed, several works have contributed to our understanding here, including (Shamir, 2018; Bartlett et al., 2019; Du & Hu, 2019; Wu et al., 2019; Hu et al., 2020; Wu et al., 2019), and (Gunasekar et al., 2017; Soudry et al., 2018; Ji & Telgarsky, 2018; Arora et al., 2019a; Rahaman et al., 2018; Du et al., 2018a) in the related area of implicit regularization.

For our purposes, (Arora et al., 2018a) exemplifies the current state of the art and its shortcomings. Loosely speaking, Theorem 1 in (Arora et al., 2018a) states that, when the initial loss is *small*, gradient flow (13) solves problem (6) to an accuracy of $\epsilon > 0$ in the order of

$$C^{-(1-\frac{1}{N})} \log(1/\epsilon) \qquad (29)$$

time units, where C is independent of the depth N of the linear network. For completeness, Theorem 1 in (Arora et al., 2018a) is reviewed in Appendix K.

Theorem 1 in (Arora et al., 2018a) might disappoint the researchers. For one, (29) suggests that increasing the network depth N only marginally speeds up the training.

More concerning is that Theorem 1 in (Arora et al., 2018a) requires a close initialization, which is *not* necessary for convergence, see Theorem 3.8. Indeed, Theorem 1 in (Arora et al., 2018a) hinges on a perturbation argument, whereby the initialization $W_{\mathbf{N},0}$ of gradient flow (13) must satisfy

$$L_N(W_{\mathbf{N},0}) = \text{sufficiently small.} \qquad \text{(see (6))} \qquad (30)$$

In this sense, (Arora et al., 2018a) joins the growing body of literature that quantifies the behavior of neural networks when the *learning trajectory is short* (Du et al., 2018b; Li & Liang, 2018; Allen-Zhu et al., 2018b;a; Zou et al., 2018; Arora et al., 2019b; Tian, 2017; Brutzkus & Globerson, 2017; Brutzkus et al., 2017; Du & Lee, 2018; Zhong et al., 2017; Zhang et al., 2018; Wu et al., 2019; Shin & Karniadakis, 2019; Zhang et al., 2019; Su & Yang, 2019; Cao & Gu, 2019; Chen et al., 2019; Oymak & Soltanolkotabi, 2018), to name a few.

To be sure, restricting the initialization is necessary for successful training (Sutskever et al., 2013). For example, gradient flow would stall when initialized at a saddle point.

However, it is widely-believed that first-order algorithms can successfully train neural networks far beyond the lazy training regime considered by (Arora et al., 2018a) and others, and the line of research exemplified by (Arora et al., 2018a) is, while valuable, highly over-represented in the literature.

Indeed, the learning trajectory of neural networks is in general *not short*, and the learning is often *not local*. We refer to (Chizat et al., 2019; Yehudai & Shamir, 2019) for a detailed critique of lazy training, see also Appendix K.

Let us dub this more general regime *non-local training*. Quantifying the non-local convergence rate of linear networks is a vital step towards understanding the non-local training of neural networks in general.

In an important setting, this section indeed quantifies the non-local training of linear networks, and addresses both of the shortcomings of Theorem 1 in (Arora et al., 2018a).

More specifically, for the case $r = 1$ in (9), Theorem 4.4 below quantifies the convergence rate of gradient flow (13) to a solution of problem (6), even when (30) is *violated*.

Moreover, Theorem 4.4 establishes that the faraway convergence rate of gradient flow improves by increasing the network depth. All assumptions for this section are collected in Assumption 3.6. Let us turn to the details now.

Instead of the convergence rate of gradient flow (13) to a solution of problem (6), we equivalently study the convergence rate of induced flow (17), as detailed next. The following result is a consequence of Theorem 3.8, proved in Appendix L.

Lemma 4.1 (Convergence of induced flow)**.** *In the setting of Theorem 3.8, if gradient flow (13) converges to a solution of problem (6), then induced flow (17) converges to the solution Z_1 of problem (24). Here, Z_1 was defined in (27).*

To quantify the convergence rate of induced flow (17), let us define the new loss function

$$L_{1,1}(W) := \frac{1}{2}\|Z_1 - W\|_F^2. \qquad \text{(see (27))} \qquad (31)$$

In this section, we often opt for subscripts to compactly show the dependence of variables on time t, for example, W_t as a shorthand for the induced flow $W(t)$. With $r = 1$, recall that induced flow (17) satisfies $\text{rank}(W_t) \le 1$, see (9,18). Assuming that $\text{rank}(W_0) = 1$ at initialization, the induced flow remains rank-1 by Lemma 3.3. Recall also the analytic SVD of the induced flow in (19). The induced flow thus admits the analytic thin SVD

$$W_t \overset{\text{tSVD}}{=} u_t \cdot s_t \cdot v_t^\top, \qquad \forall t \ge 0, \qquad (32)$$

where $u_t \in \mathbb{R}^{d_y}$ and $v_t \in \mathbb{R}^{d_x}$ have unit-norm, and $s_t > 0$ is the only nonzero singular value of W_t.

A simple calculation using (32), deferred to Appendix M, upper bounds the loss function $L_{1,1}$ in (31) as

$$L_{1,1}(W_t) \le \overbrace{\frac{1}{2}(s_t - u_t^\top Z_1 v_t)^2}^{T_{1,t}} + s_Z \underbrace{(s_Z - u_t^\top Z_1 v_t)}_{T_{2,t}}. \qquad (33)$$

Roughly speaking, $T_{1,t}$ above gauges the error in estimating the (only) nonzero singular value s_Z of the target Z_1, whereas $T_{2,t}$ gauges the misalignment between W_t and Z_1. Both $T_{1,t}, T_{2,t}$ are nonnegative for all $t \ge 0$, see (27,32).

To quantify the convergence rate of induced flow (17) to the global minimizer Z_1 of problem (24), we next write down the evolution of the loss function $L_{1,1}$ in (31) as

$$\begin{aligned} &\frac{\mathrm{d}L_{1,1}(W_t)}{\mathrm{d}t} \\ &= \left\langle \nabla L_{1,1}(W_t), \dot{W}_t \right\rangle \qquad \text{(chain rule)} \\ &= -m \left\langle W_t - Z_1, \mathcal{A}_{W_t}(W_t - Z) \right\rangle, \ \text{(see (17,25))} \qquad (34) \end{aligned}$$

where the last line also uses the whitened data in (23). Starting with the definition of $\mathcal{A}_{W_t}$ in (16), we can bound the last line of (34), see Appendix N for the proof.

Lemma 4.2 (Evolution of loss)**.** *For induced flow (17) and* *the loss function $L_{1,1}$ in (31), it holds that*

$$\begin{aligned}&\frac{dL_{1,1}(W_t)}{dt}\\&\leq -2mNs_t^{2-\frac{2}{N}}T_{1,t} - 2ms_t^{2-\frac{2}{N}}(u_t^\top Z_1 v_t)T_{2,t}\\&+\sqrt{2}mNs_t^{2-\frac{2}{N}}\gamma_Z\sqrt{T_{1,t}}T_{2,t} + 2ms_t^{2-\frac{2}{N}}s_{Z,2}T_{2,t}, \quad (35)\end{aligned}$$

see (26,32,33) for the notation involved.

Loosely speaking, the two nonpositive terms on the right-hand side of (35) are the contribution of the target matrix Z_1 in (27), whereas the two nonnegative terms there are the contribution of the residual matrix $Z - Z_1$. The (unwanted) nonnegative terms in (35) vanish if $Z = Z_1$ is rank-1 and, consequently, $\gamma_Z = s_{Z,2} = 0$, see (26). In view of (33,35), we make two observations:

① Both $T_{1,t}$ and $T_{2,t}$ in (33) appear with negative factors in the dynamics of (35). For loss $L_{1,1}$ to reduce rapidly, we must ensure that s_t and $u_t^\top Z_1 v_t$ both remain bounded away from zero for all $t \geq 0$.

② $T_{1,t}$ has a large negative factor of $-N$ in the evolution of loss function in (35), and is therefore expected to reduce much faster with time for deeper linear networks.

Let us fix $\alpha \in [\gamma_Z, 1)$ and $\beta > 1$. Given the first observation above, it is natural to restrict the initialization of gradient flow to a subset of the feasible set of problem (24), specified as

$$\begin{aligned}\mathcal{N}_{\alpha,\beta}(Z_1) := \Big\{ &W \overset{\text{tSVD}}{=} u_W \cdot s_W \cdot v_W^\top : \\ &(\alpha - \gamma_Z)s_Z < s_W < \beta s_Z, \\ &u_W^\top Z_1 v_W > \alpha s_Z \Big\} \subset \mathbb{R}^{d_y \times d_x}, \qquad (36)\end{aligned}$$

where s_Z, γ_Z were defined in (26).

The necessity of such a restriction was discussed after (28), and the (new) upper bound on s_W in (36) controls the (unwanted) positive terms in (35). Note that $\mathcal{N}_{\alpha,\beta}(Z_1)$ is a neighborhood of Z_1, i.e., $Z_1 \in \mathcal{N}_{\alpha,\beta}(Z_1)$ by (27).

Once initialized in $\mathcal{N}_{\alpha,\beta}(Z_1)$, induced flow (17) remains in $\mathcal{N}_{\alpha,\beta}(Z_1)$, see Appendix O, closely related to Lemma 3.7.

Lemma 4.3 (Stable set). *Fix $\alpha \in [\gamma_Z, 1)$ and $\beta > 1$. For induced flow (17), $W_0 \in \mathcal{N}_{\alpha,\beta}(Z_1)$ implies that $W_t \in \mathcal{N}_{\alpha,\beta}(Z_1)$ for all $t \geq 0$. Above, Assumption 3.6 and the notation therein are in force.*

In view of Lemma 4.3, we can now use (36) to bound s_t and $u_t^\top Z_1 v_t$ in (35). We can then distinguish two regimes (fast and slow convergence) in the dynamics of the loss function in (35) depending on the dominant term on the right-hand side of (33). The remaining technical details are deferred to Appendix P and we finally arrive at the following result.

Theorem 4.4 (Convergence rate). *With Assumption 3.6 and its notation in force, fix $\alpha \in (\gamma_Z, 1)$ and $\beta > 1$. Suppose that the inverse spectral gap γ_Z is small enough so that the exponents below are both negative.*

Consider gradient flow (13) with the balanced initialization $W_{\mathbf{N},0} = (W_{1,0}, \cdots, W_{N,0}) \in \mathbb{R}^{d_\mathbf{N}}$ such that $W_0 := W_{N,0} \cdots W_{1,0} \in \mathbb{R}^{d_y \times d_x}$ satisfies

$$\begin{aligned}&\mathit{rank}(W_0) = 1, \qquad W_0 \overset{\mathit{tSVD}}{=} u_0 s_0 v_0^\top, \\ &(\alpha - \gamma_Z)s_Z < s_0 < \beta s_Z, \qquad u_0^\top Z_1 v_0 > \alpha s_Z. \quad (37)\end{aligned}$$

Let $W_\mathbf{N}(t) = (W_1(t), \cdots, W_N(t))$ be the output of gradient flow (13) at time t, and set $W(t) := W_N(t) \cdots W_1(t)$, which satisfies rank$(W(t)) = 1$ for every $t \geq 0$.

Let $\tau \geq 0$ be the first time when $s(\tau) \leq \sqrt{6}s_Z$, where $s(\tau)$ is the (only) nonzero singular value of $W(\tau)$. Then the distance to the target matrix Z_1 in (27) evolves as

$$\forall t \leq \tau, \qquad \|Z_1 - W(t)\|_F^2 \leq \|Z_1 - W_0\|_F^2 \cdot e^{-mNs_Z^{2-\frac{2}{N}}\left((\alpha-\gamma_Z)^{2-\frac{2}{N}} - 2\gamma_Z\beta^{2-\frac{2}{N}}\right)t}. \quad (38a)$$

$$\forall t \geq \tau, \qquad \|Z_1 - W(t)\|_F^2 \leq \|Z_1 - W(\tau)\|_F^2 \cdot e^{-ms_Z^{2-\frac{2}{N}}\left(\alpha(\alpha-\gamma_Z)^{2-\frac{2}{N}} - 2\gamma_Z N\beta^{2-\frac{2}{N}}\right)(t-\tau)}. \quad (38b)$$

Under Assumption 3.6, Theorem 4.4 states that gradient flow successfully trains a linear network with linear rate, when initialized in the stable set.

As we will see shortly, Theorem 4.4 is the first result to quantify the convergence rate of gradient flow beyond the widely-studied lazy training regime. The remarks after Theorem 3.8 again apply here about Assumption 3.6 and the case $r > 1$. A few additional remarks are in order.

① Rephrasing (38), gradient flow (13) solves problem (6) to an accuracy of $\epsilon > 0$ in the order of

$$\begin{cases}\frac{1}{mNs_Z^2}\left((\alpha-\gamma_Z)^2 - 2\gamma_Z\beta^2\right)^{-1}\log(C/\epsilon) & \epsilon > \epsilon_0\\ \frac{1}{ms_Z^2}\left(\alpha(\alpha-\gamma_Z)^2 - 2\gamma_Z N\beta^2\right)^{-1}\log(C/\epsilon) & \\ -\tau\left(N\frac{(\alpha-\gamma_Z)^2 - 2\gamma_Z\beta^2}{\alpha(\alpha-\gamma_Z)^2 - 2\gamma_Z N\beta^2} - 1\right) & \epsilon \leq \epsilon_0\end{cases}$$

time units. Above, ϵ_0 is the right-hand side of (38b), evaluated at $t = \tau$.

② In Theorem 4.4, the end-to-end initialization matrix W_0 in (37) is positively correlated with the target matrix Z_1, and away from the origin, see our earlier discussions for the necessity of such restricted initialization. Note also that (37) should not be seen as an initialization scheme but as a theoretical result.

③ The faraway convergence rate in (38) improves with increasing the network depth N, whereas the nearby convergence rate does not appear to benefit from increasing

N. This improved faraway convergence rate should be contrasted with Arora's result in (29).

④ Crucially, the lazy training results fail to apply here. To see this, with the initialization $W_{\mathbf{N},0} = (W_{1,0}, \cdots, W_{N,0})$, Claim 1 in (Arora et al., 2018a) uses a perturbation argument, which requires that

$$\|Z - W_{N,0} \cdots W_{1,0}\|_F < s_{\min}(Z), \tag{39}$$

which is *impossible* unless trivially $\mathrm{rank}(Z) \le r$. Indeed, the network architecture forces that $\mathrm{rank}(W_{N,0} \cdots W_{1,0}) \le r$, see (9).

In contrast, Theorem 4.4 applies even when (39) is violated, as it does away entirely with the limitations of a perturbation argument. Theorem 4.4 thus ventures beyond the reach of the lazy training regime in (Arora et al., 2018a), which has dominated the recent literature of neural networks, thus signifying the importance of this breakthrough.

Thorough numerics for linear networks are abound, see for example (Bah et al., 2019; Arora et al., 2018a;b), and we refrain from lengthy simulations and only provide a numerical example in Figure 1. to visualize the (gradual) change of regimes from fast to slow convergence, see (38a,38b). This example also suggests new research questions about linear networks.

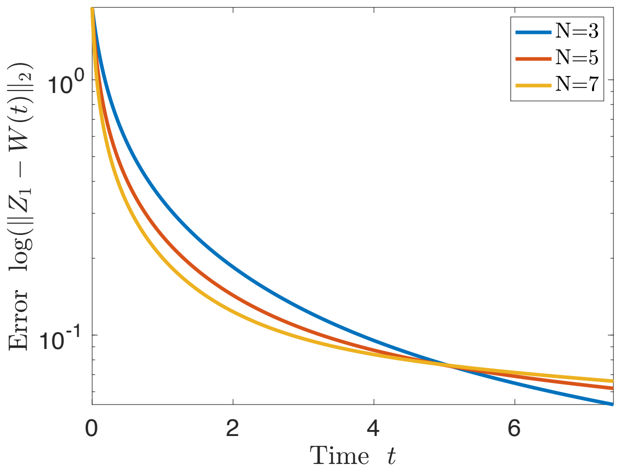

Figure 1. Suppose that the sample size is $m = 50$, and consider a randomly-generated whitened training dataset

$$(X, Y) \in \mathbb{R}^{d_x \times m} \times \mathbb{R}^{d_y \times m},$$

with $d_x = 5$ and $d_y = 1$. For this dataset, the above figure depicts the distance from induced flow (17) to the target vector $Z_1 = Z = YX^\top/m$ in (25,27), plotted versus time t, for training a linear network with d_x inputs and d_y outputs, as the network depth N varies.

The direction of the initial end-to-end vector $W_0 \in \mathbb{R}^{d_y \times d_x}$ is obtained by randomly rotating the direction of the target vector Z_1 by about 30 degrees. We also set $\|W_0\|_2 = 10\|Z\|_2$. Instead of induced flow (17), we implemented the discretization of (17) obtained from the explicit (or forward) Euler method with a step size of 10^{-6} with 10^5 steps.

This simple numerical example visualizes the (gradual) slow-down in the convergence rate of gradient flow with time, see (38), and also shows the faster faraway convergence rate for deeper networks, see Theorem 4.4.

The above figure also suggests that the nearby convergence rate of gradient flow (13) might actually be slower for deeper networks. It is however difficult to theoretically infer this from Theorem 4.4, because (38b) is an *upper bound* for the nearby error. The precise nearby convergence rates of linear networks (and any trade-offs associated with the network depth) thus remain as open questions. Note also that the local analysis of (Arora et al., 2018a) cannot be applied here, as discussed after Theorem 4.4.

References

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv preprint arXiv:1811.04918*, 2018a.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018b.

Arora, S., Cohen, N., Golowich, N., and Hu, W. A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*, 2018a.

Arora, S., Cohen, N., and Hazan, E. On the optimization of deep networks: Implicit acceleration by overparameterization. *arXiv preprint arXiv:1802.06509*, 2018b.

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization. *arXiv preprint arXiv:1905.13655*, 2019a.

Arora, S., Du, S. S., Hu, W., Li, Z., and Wang, R. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019b.

Bah, B., Rauhut, H., Terstiege, U., and Westdickenberg, M. Learning deep linear neural networks: Riemannian gradient flows and convergence to global minimizers. *arXiv preprint arXiv:1910.05505*, 2019.

Baldi, P. and Hornik, K. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

Bartlett, P. L., Helmbold, D. P., and Long, P. M. Gradient descent with identity initialization efficiently learns positive-definite linear transformations by deep residual networks. *Neural computation*, 31(3):477–502, 2019.

Berthet, Q., Rigollet, P., et al. Optimal detection of sparse principal components in high dimension. *The Annals of Statistics*, 41(4):1780–1815, 2013.

Brutzkus, A. and Globerson, A. Globally optimal gradient descent for a convnet with gaussian inputs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 605–614. JMLR. org, 2017.

Brutzkus, A., Globerson, A., Malach, E., and Shalev-Shwartz, S. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*, 2017.

Cao, Y. and Gu, Q. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 10835–10845, 2019.

Chen, Z., Cao, Y., Zou, D., and Gu, Q. How much over-parameterization is sufficient to learn deep relu networks? *arXiv preprint arXiv:1911.12360*, 2019.

Chitour, Y., Liao, Z., and Couillet, R. A geometric approach of gradient descent algorithms in neural networks. *arXiv preprint arXiv:1811.03568*, 2018.

Chizat, L., Oyallon, E., and Bach, F. On lazy training in differentiable programming. 2019.

Deshpande, Y. and Montanari, A. Information-theoretically optimal sparse pca. In *2014 IEEE International Symposium on Information Theory*, pp. 2197–2201. IEEE, 2014.

Du, S. S. and Hu, W. Width provably matters in optimization for deep linear neural networks. *arXiv preprint arXiv:1901.08572*, 2019.

Du, S. S. and Lee, J. D. On the power of over-parametrization in neural networks with quadratic activation. *arXiv preprint arXiv:1803.01206*, 2018.

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

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018b.

Eckart, C. and Young, G. The approximation of one matrix by another of lower rank. *Psychometrika*, 1:211–218, 1936. doi: 10.1007/BF02288367.

Eftekhari, A., Hauser, R., and Grammenos, A. Moses: A streaming algorithm for linear dimensionality reduction. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

Ge, R., Huang, F., Jin, C., and Yuan, Y. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pp. 797–842, 2015.

Golub, G. H., Hoffman, A., and Stewart, G. W. A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and its applications*, 88:317–327, 1987.

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

Hardt, M. and Ma, T. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

Hauser, R. A. and Eftekhari, A. Pca by optimisation of symmetric functions has no spurious local optima. *arXiv preprint arXiv:1805.07459*, 2018.

Hauser, R. A., Eftekhari, A., and Matzinger, H. F. Pca by determinant optimisation has no spurious local optima. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1504–1511, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Helmke, U. and Shayman, M. A. Critical points of matrix least squares distance functions. *Linear Algebra and its Applications*, 215:1–19, 1995.

Hu, W., Xiao, L., and Pennington, J. Provable benefit of orthogonal initialization in optimizing deep linear networks. *arXiv preprint arXiv:2001.05992*, 2020.

Illashenko and Yakovenko. *Lectures on analytic differential equations*, volume 86. American Mathematical Soc., 2008.

Ji, Z. and Telgarsky, M. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2018.

Johnstone, I. M. et al. On the distribution of the largest eigenvalue in principal components analysis. *The Annals of statistics*, 29(2):295–327, 2001.

Kawaguchi, K. Deep learning without poor local minima. In *Advances in neural information processing systems*, pp. 586–594, 2016.

Laurent, T. Deep linear networks with arbitrary loss: All local minima are global. 2018.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

Lee, J. D., Panageas, I., Piliouras, G., Simchowitz, M., Jordan, M. I., and Recht, B. First-order methods almost always avoid saddle points. *arXiv preprint arXiv:1710.07406*, 2017.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pp. 8157–8166, 2018.

Lu, H. and Kawaguchi, K. Depth creates no bad local minima. *arXiv preprint arXiv:1702.08580*, 2017.

Mackey, L. W. Deflation methods for sparse pca. In *Advances in neural information processing systems*, pp. 1017–1024, 2009.

Mirsky, L. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford*, pp. 1156–1159, 1966.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Nguyen, Q. On connected sublevel sets in deep learning. *arXiv preprint arXiv:1901.07417*, 2019.

Oymak, S. and Soltanolkotabi, M. Overparameterized nonlinear learning: Gradient descent takes the shortest path? *arXiv preprint arXiv:1812.10004*, 2018.

Parks, H. R. and Krantz, S. *A primer of real analytic functions*. Birkhäuser Verlag Boston (MA), 1992.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. On the spectral bias of neural networks. *arXiv preprint arXiv:1806.08734*, 2018.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Shamir, O. Exponential convergence time of gradient descent for one-dimensional deep linear neural networks. *arXiv preprint arXiv:1809.08587*, 2018.

Shin, Y. and Karniadakis, G. E. Trainability and data-dependent initialization of over-parameterized relu neural networks. *arXiv preprint arXiv:1907.09696*, 2019.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Su, L. and Yang, P. On learning over-parameterized neural networks: A functional approximation prospective. *arXiv preprint arXiv:1905.10826*, 2019.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147, 2013.

Tian, Y. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3404–3413. JMLR. org, 2017.

Townsend, J. Differentiating the singular value decomposition. Technical report, Technical report, 2016.

Trager, M., Kohn, K., and Bruna, J. Pure and spurious critical points: a geometric study of linear networks. *arXiv preprint arXiv:1910.01671*, 2019.

Vershynin, R. How close is the sample covariance matrix to the actual covariance matrix? *Journal of Theoretical Probability*, 25(3):655–686, 2012.

Wu, X., Du, S. S., and Ward, R. Global convergence of adaptive gradient methods for an over-parameterized neural network. *arXiv preprint arXiv:1902.07111*, 2019.

Yan, W.-Y., Helmke, U., and Moore, J. B. Global analysis of oja's flow for neural networks. *IEEE Transactions on Neural Networks*, 5(5):674–683, 1994.

Yehudai, G. and Shamir, O. On the power and limitations of random features for understanding neural networks. *arXiv preprint arXiv:1904.00687*, 2019.

Yun, C., Sra, S., and Jadbabaie, A. Global optimality conditions for deep neural networks. *arXiv preprint arXiv:1707.02444*, 2017.

Zhang, G., Martens, J., and Grosse, R. Fast convergence of natural gradient descent for overparameterized neural networks. *arXiv preprint arXiv:1905.10961*, 2019.

Zhang, X., Yu, Y., Wang, L., and Gu, Q. Learning one-hidden-layer relu networks via gradient descent. *arXiv preprint arXiv:1806.07808*, 2018.

Zhang, Y. and Ghaoui, L. E. Large-scale sparse principal component analysis with application to text data. In *Advances in Neural Information Processing Systems*, pp. 532–539, 2011.

Zhong, K., Song, Z., and Dhillon, I. S. Learning non-overlapping convolutional neural networks with multiple kernels. *arXiv preprint arXiv:1711.03440*, 2017.

Zhu, Z., Soudry, D., Eldar, Y. C., and Wakin, M. B. The global optimization geometry of shallow linear neural networks. *Journal of Mathematical Imaging and Vision*, pp. 1–14, 2019.

Zou, D., Cao, Y., Zhou, D., and Gu, Q. Stochastic gradient descent optimizes over-parameterized deep relu networks. *arXiv preprint arXiv:1811.08888*, 2018.