A Network Perspective on Gradient Flow Equations for Deep Linear Neural Networks

Joel Wendin and Claudio Altafini
Department of Electrical Engineering,
Linköping University, SE-58183 Linköping, Sweden

Deep learning has over the last decade proven to be a powerful method to solve complex tasks in computer vision, natural language processing and chemistry. At the same time, understanding of the theory behind these successes has been progressing more slowly. Even though the training of deep neural networks is typically done via simple first order optimization methods on highly non-convex objectives, bad local minima are often avoided and the networks generalize to unseen data. A general difficulty in understanding the training dynamics of neural networks is due to their nonlinear nature. This work aims at investigating the recent developments in the study of *deep linear neural networks*, which have been used as a simplified but analytically treatable proxy for nonlinear neural networks. Deep linear networks have no activation functions and hence lose the expressive power of their nonlinear counterparts, but they still retain some interesting properties, such as the nonlinearity of the training dynamics, the non-convex objective (loss function), and the overparameterization. They are also trained using the same algorithm as nonlinear neural networks. Our contribution is to treat the network in a graphical framework and to reformulate the system dynamics in terms of its adjacency matrix, which simplifies and provides insight into the system properties.

A linear neural network $f : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ is a function parameterized by weight matrices W_1,...,W_h, $W_j \in \mathbb{R}^{d_j \times d_{j-1}}$ that define a linear map $f(x; \{W_j\}) = W_h \cdots W_1 x$. The network connects the input to the output via $h-1$ hidden layers of sizes $d_1, ..., d_{h-1}$ (Fig. 1(a)), and the (k, ℓ)th element of W_j defines the weight of the edge from the ℓth node in the $(j-1)$th layer to the kth node in the jth layer. We study how the network learns these weights when trained on a linear regression task, i.e., minimizing a quadratic objective. This objective can be elegantly formulated in terms of the network adjacency matrix, A, and is a p.s.d. Lyapunov function, $\mathcal{L}(A)$.

Neural networks are commonly trained using gradient descent methods, where the parameters of the model are updated by taking a step in the opposite direction of the gradient of the objective: $A(t+1) = A(t) - \eta \nabla_A \mathcal{L}(A(t))$. In deep linear networks, we can study these dynamics in continuous time by letting the step size shrink to zero, obtaining the so-called gradient flow equations $\dot{A} = -\nabla_A \mathcal{L}$ [4]. For shallow networks, $h = 1$, $\mathcal{L}$ is convex, the dynamics are linear, and trajectories always converge to the unique global optimum. As soon as $h > 1$, $\mathcal{L}$ is non-convex, the dynamics are nonlinear, and there are infinitely many (possibly unbounded) optimal solutions. However, despite this, deep linear networks are rather well-behaved. One reason is that they obey a set of conservation laws which ensure that any trajectory is bounded and converges to a critical point of the Lyapunov derivative $\dot{\mathcal{L}} = \langle \dot{A}, \nabla_A \mathcal{L} \rangle_F = -\|\dot{A}\|_F^2$, and thus to a critical point of $\dot{A}$ [2].

Another interesting property of deep linear networks is that all local minima are global

minima, while all other critical points are saddles [3]. In addition to having provided a parameterization of all critical points of $\dot{A}$, the work [1] investigated the second order nature of saddle points and derived a rank condition on the weight matrices to distinguish between strict and non-strict saddles of $\dot{A}$. Non-strict saddles are characterized by the Hessian lacking negative eigenvalues, and trajectories can be significantly slowed down when passing close by. A sufficient condition for a saddle A^* to be non-strict is that three of its weight matrices have minimal rank $r = \text{rank}((A^*)^h)$, and that the input-output map is the optimal rank-r solution: $\mathcal{L}(A^*) = \min_{A,\text{rank}(A^h)=r} \mathcal{L}(A)$.

Although a deep linear network almost always converges to a global optimum, qualitatively it behaves differently depending on whether it is initialized near or far away from the origin. For initializations close enough to the origin, the network learns the data modes sequentially from largest to smallest, which is visible as sudden decreases in the objective $\mathcal{L}$ (Fig. 1(b)). We can understand the phases where $\mathcal{L}$ is almost constant as an effect of the trajectory passing close by non-strict saddle points. In contrast, when initializing far away from the origin (Fig. 1(c)) the system avoids the non-strict saddles: convergence is faster, and all modes are learnt simultaneously.

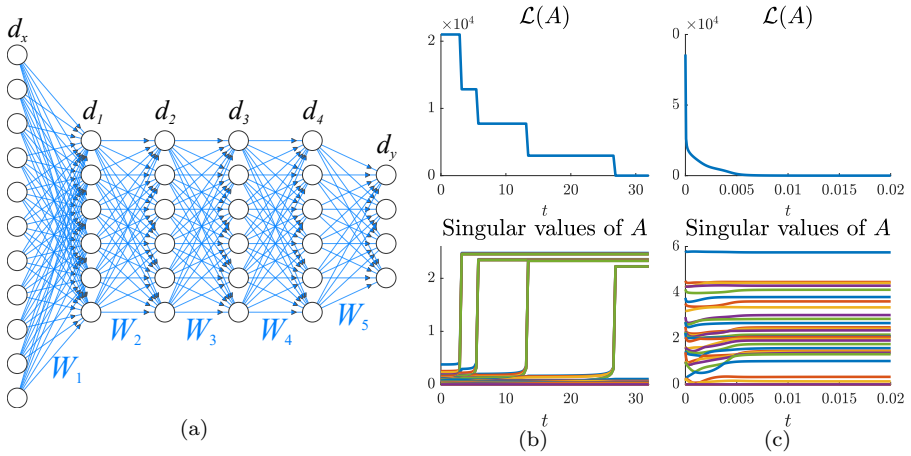

Figure 1: (a) A linear neural network ($h = 5$, $d_x = 11$, $d_i = 6$, $d_y = 4$); its corresponding loss function $\mathcal{L}$ and singular values of A during training when initializing (b) near the origin; and (c) far from the origin.

References

[1] E. M. Achour, F. Malgouyres, and S. Gerchinovitz. The loss landscape of deep linear neural networks: a second-order analysis. *Journal of Machine Learning Research*, 25(242):1–76, 2024.

[2] Y. Chitour, Z. Liao, and R. Couillet. A geometric approach of gradient descent algorithms in linear neural networks. *Mathematical Control and Related Fields*, 13(3):918–945, 2023.

[3] K. Kawaguchi. Deep learning without poor local minima. *Advances in neural information processing systems*, 2016.

[4] A. Saxe, J. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Represenatations*, 2014.