A. Numerical Verification

The scale of the linear regression problem we consider in the numerical section is $D = 400$, $n = 100$, and $m = 1$.

A.1. Convergence of single-hidden-layer linear network via imbalanced initialization

Generating training data The synthetic training data is generated as following:

1) For data matrix X, first we generate $X_0 \in \mathbb{R}^{n \times D}$ with all the entries sampled from $\mathcal{N}(0, 1)$, and take its SVD $X_0 = W\Sigma^{1/2}\Phi_1$. Then we let $X = W\Phi_1$, hence we have all the singular values of X being 1. Here $r = \text{rank}(X) = n = 100$.

2) For Y, we first sample $\Theta \sim \mathcal{N}(0, D^{-1}I_D)$, and $\epsilon \sim \mathcal{N}(0, 0.01^2 I_n)$, then we let $Y = X\Theta + \epsilon$.

Initialization and Training We set the hidden layer width $h = 500$. We initialize $U(0), V(0)$ with

$$U(0) = \sigma_U U_0,\ V(0) = \sigma_V V_0, \qquad [U_0]_{ij}, [V_0]_{ij} \overset{i.i.d.}{\sim} \mathcal{N}(0, 1)\,.$$

and we consider three cases of such initialization: 1) $\sigma_U = 0.1,\ \sigma_V = 0.1$; 2) $\sigma_U = 0.5,\ \sigma_V = 0.02$; 3) $\sigma_U = 0.05,\ \sigma_V = 0.2$. Such setting ensures the initial end-to-end function are identical for all cases but with different levels of imbalance. For these three cases, we run gradient descent on the averaged loss $\tilde{L} = \frac{1}{n}\|Y - XUV^T\|_F^2$ with step size[2] $\eta = 5e - 4$.

For comparison, we also consider the balanced initialization that corresponds to the same end-to-end matrix. For a given $\Theta(0) = U(0)V^T(0)$, we choose an arbitrary $Q \in \mathbb{R}^{h \times m}$ with $Q^T Q = I_m$, then a balanced initialization is given by

$$U_{\text{balanced}}(0) = \Theta(0)\left[\Theta^T(0)\Phi_1\Phi_1^T\Theta(0)\right]^{-1/4} Q^T, \quad V_{\text{balanced}}(0) = \left[\Theta^T(0)\Phi_1\Phi_1^T\Theta(0)\right]^{1/4} Q\,.$$

Such initialization ensures the imbalanced is the zero matrix while keeping the end-to-end matrix as $\Theta(0)$. We note here the choice of Q does not affect the error trajectory $E(t)$, hence the loss $\mathcal{L}(t)$.

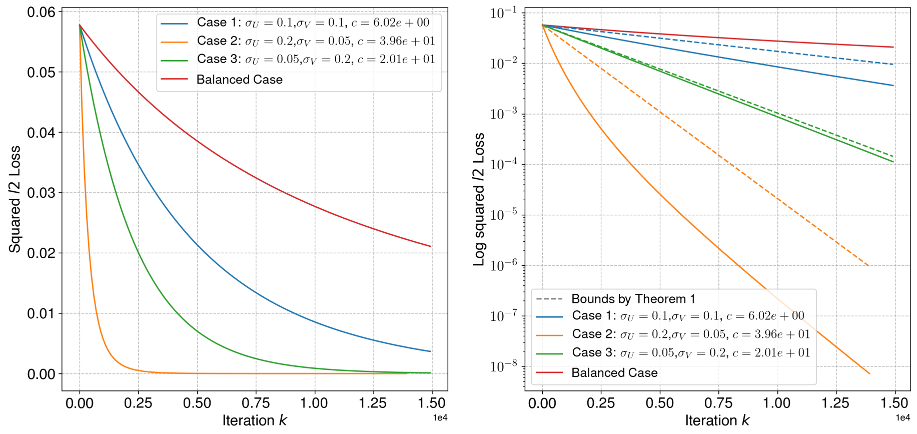

Figure 3. Convergence of gradient descent with different initial level of imbalance, $c := \lambda_r(\Lambda(0)) + \lambda_m(-\Lambda(0))$.

From Fig.3, we see that given fixed step size, the convergence rate is improved as we increase the level of the imbalance at initialization and the balanced initialization is the slowest among all cases. Notably, in case 3, our bound is almost tight. For case 2, our bound is tight in characterizing the asymptotic rate of convergence. Our analysis does not provide convergence guarantee for the balanced case, while Arora et al. (2018a;b) have shown linear convergence for certain cases with zero imbalance at initialization. This suggests the need of a unified convergence analysis, that is applicable for both balanced and imbalanced initialization, to obtain tighter convergence guarantees. This is subject of current research.

A.2. Implicit regularization on wide single-hidden-layer linear network

Generating training data The synthetic training data is generated as following:

[2]To compute the bound from Theorem 1, the step size is scaled by $n/2$ to account for that the gradient descent uses rescaled loss function.

1) For data matrix X, first we generate $X \in \mathbb{R}^{n\times D}$ with all the entries sampled from $\mathcal{N}(0, D^{-1})$;

2) For Y, we first sample $\Theta \sim \mathcal{N}(0, D^{-1} I_D)$, and $\epsilon \sim \mathcal{N}(0, 0.01^2 I_n)$, then we let $Y = X\Theta + \epsilon$.

Initialization and Training We initialize $U(0), V(0)$ with $[U(0)]_{ij} \sim \mathcal{N}(0, h^{-1})$, $[V(0)]_{ij} \sim \mathcal{N}(0, h^{-1})$ and run gradient descent on the averaged loss $\tilde{L} = \frac{1}{n}\|Y - XUV^T\|_F^2$ with step size $\eta = 5e-3$. The training stops when the loss is below $1e-8$. We run the algorithm for various h from 500 to 10000, and we repeat 5 runs for each h.

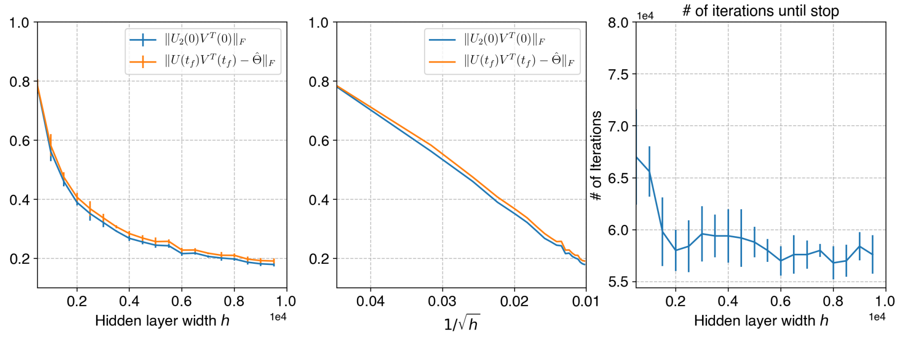

Figure 4. Implicit bias of wide single-hidden-layer linear network under random initialization. The line is plotting the average over 5 runs for each h, and the error bar shows the standard deviation. The gradient descent stops at iteration t_f.

Fig.4 clearly shows that the distance between the trained network and the min-norm solution, $\|U(t_f)V^T(t_f) - \hat{\Theta}\|_F$, decreases as the width h increases and the middle plot verifies the asymptotic rate $\mathcal{O}(h^{-1/2})$.

B. Comparison with the NTK Initialization for wide single-hidden-layer linear networks

In Section 3.2, we analyzed implicit bias of wide single-hidden-layer linear networks under properly scaled random initialization. Our initialization for network weights U, V is different from the typical setting in previous works (Jacot et al., 2018; Du & Hu, 2019; Arora et al., 2019c). In this section, we show that under our setting, the gradient flow is related to the NTK flow by 1) reparametrization and rescaling in time ; 2) proper scaling of the network output. The use of output scaling is also used in Arora et al. (2019c).

In this paper we work with a single-hidden-layer linear network defined as $f : \mathbb{R}^D \to \mathbb{R}^m$, $f(x; V, U) = VU^T x$, which is parametrized by U, V. Then we analyze the gradient flow on the loss function $\mathcal{L}(V, U) = \frac{1}{2}\left\|Y - XUV^T\right\|_F^2$, given the data and output matrix X, Y. Lastly, in Section 4.2, we initialize $U(0), V(0)$ such that all the entries are randomly drawn from $\mathcal{N}\left(0, h^{-2\alpha}\right)$ $(1/4 < \alpha \leq 1/2)$, where h is the hidden layer width.

Now we define $\tilde{U} := h^\alpha U, \tilde{V} := h^\alpha V$, then the loss function can be written as

$$\begin{aligned}\mathcal{L}(V,U) = \tilde{\mathcal{L}}(\tilde{V},\tilde{U}) = \frac{1}{2}\left\|Y - \frac{1}{h^{2\alpha}}X\tilde{U}\tilde{V}^T\right\|_F^2 &= \frac{1}{2}\left\|Y - \frac{\sqrt{m}}{h^{2\alpha-\frac{1}{2}}}\frac{1}{\sqrt{mh}}X\tilde{U}\tilde{V}^T\right\|_F^2 \\ &= \frac{1}{2}\sum_{i=1}^n \left\|y^{(i)} - \frac{\sqrt{m}}{h^{2\alpha-\frac{1}{2}}}\frac{1}{\sqrt{mh}}\tilde{V}\tilde{U}^T x^{(i)}\right\|_2^2 \\ &:= \sum_{i=1}^n \left\|y^{(i)} - \frac{\sqrt{m}}{h^{2\alpha-\frac{1}{2}}}\tilde{f}(x;\tilde{V},\tilde{U})\right\|_2^2\end{aligned}$$

Notice that $\tilde{f}(x; \tilde{V}, \tilde{U}) = \frac{1}{\sqrt{mh}}\tilde{V}\tilde{U}^T x$ is the typical network discussed in previous works (Jacot et al., 2018; Du & Hu, 2019; Arora et al., 2019c). When all the entries of $U(0), V(0)$ are initialized randomly as $\mathcal{N}\left(0, h^{-2\alpha}\right)$, the entries of $\tilde{U}(0), \tilde{V}(0)$ are random samples from $\mathcal{N}(0, 1)$, which is the typical choice of initialization for NTK analysis.

However, the difference is that $\tilde{f}(x; \tilde{V}, \tilde{U})$ is scaled by $\frac{\sqrt{m}}{h^{2\alpha-\frac{1}{2}}}$. In previous work showing non-asymptotic bound between wide neural networks and its infinite width limit (Arora et al., 2019c, Theorem 3.2), the wide neural network is scaled by a small constant κ such that the prediction by the trained network is within ϵ-distance to the one by the kernel predictor of its NTK. Moreover, Arora et al. (2019c) suggests $\frac{1}{\kappa}$ should scale as $poly(\frac{1}{\epsilon})$, i.e., to make sure the trained network is arbitrarily close to the kernel predictor, κ should be vanishingly small. In our setting, the random initialization implicitly enforces such a vanishing scaling $\frac{\sqrt{m}}{h^{2\alpha-\frac{1}{2}}}$, as the width of network increases.

Lastly, we show that the gradient flow on $\mathcal{L}(V,U)$ only differs from the flow on $\tilde{\mathcal{L}}(\tilde{V},\tilde{U})$ by the time scale.

Suppose U,V[3] follows the gradient flow on $\mathcal{L}(V,U)$, we have

$$\begin{aligned} -\frac{1}{h^\alpha}\frac{\partial}{\partial U}\mathcal{L}(V,U) &= -\frac{1}{h^\alpha}X^T(Y-XUV^T)V \\ &= -\frac{1}{h^{2\alpha}}X^T\left(Y-\frac{1}{h^{2\alpha}}X\tilde{U}\tilde{V}^T\right)\tilde{V} = -\frac{\partial}{\partial\tilde{U}}\tilde{\mathcal{L}}(\tilde{V},\tilde{U})\,, \end{aligned} \tag{21}$$

and

$$\begin{aligned} -\frac{1}{h^\alpha}\frac{\partial}{\partial V}\mathcal{L}(V,U) &= -\frac{1}{h^\alpha}(Y-XUV^T)^TXU \\ &= -\frac{1}{h^{2\alpha}}\left(Y-\frac{1}{h^{2\alpha}}X\tilde{U}\tilde{V}^T\right)^TX\tilde{U} = -\frac{\partial}{\partial\tilde{V}}\tilde{\mathcal{L}}(\tilde{V},\tilde{U})\,. \end{aligned} \tag{22}$$

Consider the gradient flow on $\mathcal{L}(V,U)$ w.r.t. time t, from (21), we have

$$\begin{aligned} &\frac{d}{dt}U(t) = -\frac{\partial}{\partial U}\mathcal{L}(V(t),U(t)) \\ \Leftrightarrow\ &\frac{1}{h^\alpha}\frac{d}{dt}\tilde{U}(t) = -\frac{\partial}{\partial U}\mathcal{L}(V(t),U(t)) \\ \Leftrightarrow\ &\frac{1}{h^\alpha}\frac{d}{dt}\tilde{U}(t) = -h^\alpha\frac{\partial}{\partial\tilde{U}}\tilde{\mathcal{L}}(\tilde{V}(t),\tilde{U}(t)) \\ \Leftrightarrow\ &\frac{d}{dt}\tilde{U}(t) = -h^{2\alpha}\frac{\partial}{\partial\tilde{U}}\tilde{\mathcal{L}}(\tilde{V}(t),\tilde{U}(t))\,, \end{aligned} \tag{23}$$

Similarly from (22) we have

$$\frac{d}{dt}V(t) = -\frac{\partial}{\partial V}\mathcal{L}(V(t),U(t)) \Leftrightarrow \frac{d}{dt}\tilde{V}(t) = -h^{2\alpha}\frac{\partial}{\partial\tilde{V}}\tilde{\mathcal{L}}(\tilde{V}(t),\tilde{U}(t))\,. \tag{24}$$

From (23) and (24) we know that the gradient flow on $\mathcal{L}(V,U)$ w.r.t. time t essentially runs the gradient flow on $\tilde{\mathcal{L}}(\tilde{V},\tilde{U})$ with an scaled-up rate by $h^{2\alpha}$.

[3]We write $U(t),V(t)$ as U,V for simplicity. Same for $\tilde{U}(t),\tilde{V}(t)$.