Training and Generalization Errors for Underparameterized Neural Networks

Daniel Martin Xavier, Ludovic Chamoin, Laurent Fribourg

Training and Generalization Errors for Underparameterized Neural Networks

Daniel Martin Xavier, Ludovic Chamoin, Laurent Fribourg

***Abstract*— It has been theoretically explained, through the notion of Neural Tangent Kernel, why the training error of overparameterized networks converges linearly to 0. In this work, we focus on the case of small (or underparameterized) networks. An advantage of small networks is that they are faster to train while retaining sufficient precision to perform useful tasks in many applications. Our main theoretical contribution is to prove that the training error of small networks converges linearly to a (non-null) constant, of which we give a precise estimate. We verify this result on a neural network of 10 neurons simulating a Model Predictive Controller. We also observe that an upper bound of the generalization error follows a double-peak curve as the number of training data increases.**

***Index Terms*— Neural networks, data driven control, optimization.**

I. Introduction

RECENTLY, the notion of Neural Tangent Kernel (NTK) matrix was introduced [1], and has provided an elegant explanation of the linear convergence of the training error towards 0 on overparameterized NNs (see [2], [3]). In this type of network, the number m of parameters is much larger than the number n of data ($m \gg n$). The explanation of the convergence lies in the fact that, during the Gradient Descent (GD) procedure, the $n \times n$ NTK matrix stays close to its infinite limit, with all eigenvalues positive.

In this context, Jerray et al. [4] proposed some general assumptions ($\{$C1, C2, C3$\}$) for the convergence of the training error to 0 without reference to the number of parameters m or data n. In particular, Assumption C2 corresponds to the property of positive-definiteness of the NTK matrix, which is satisfied in the case of overparameterized networks.

In the present work, we relax Assumption C2 as Assumption C2' which states that the NTK matrix is positive semi-definite (and not positive definite): the nullspace $\mathcal{N}$ is no longer reduced to 0, but is a space of dimension larger than 1. Under Assumption C2, Jerray et al. [4] proved that the training error $\|\boldsymbol{v}(t)\|$ converges to 0 (see Theorem 2). In most cases, however, underparameterized NN do not satisfy Assumption C2 but satisfy C2'. Under the weaker Assumption C2', Theorem 2 does not hold anymore, and we prove in this work that the limit superior of $\|\boldsymbol{v}(t)\|$ is upper bounded by a positive constant b (see Theorem 3).

More precisely, the training error $\boldsymbol{v}(t)$ is now projected as $\boldsymbol{v}^2(t)$ onto $\mathcal{V}^2(t) := \mathcal{N}(t)$ and as $\boldsymbol{v}^1(t)$ onto the orthogonal space $\mathcal{V}^1(t) := \mathcal{N}^\perp(t)$. We will show that the training error converges linearly to a positive constant $b = \|\boldsymbol{v}^2(t_0)\|$ which corresponds to the projection of the error $\|\boldsymbol{v}(t_0)\|$ where t_0 is a short time of stabilization of the error after initialization.

Contribution

We show that for underparameterized NNs:

1) The *training error* converges to a constant $b = \|\boldsymbol{v}^2(t_0)\|$ (see Theorem 3).
2) This theoretical result is illustrated by Example 1, which shows that an underparameterized network accurately simulates a Model Predictive Controller (MPC).
3) We also observe on Example 2 that an upper bound of the *generalization error* (built upon some results of [5]) follows a double-peak curve as n increases, in line with with recent works on the shape of learning curves (see [6], Fig. 1 (middle) and [7], Section 6.2).

Comparison with related work

The present work extends the analysis of Jerray et al. [4] by considering the more general case where the NTK matrix is positive semi-definite (Assumption C2') instead of positive-definite (Assumption C2). As mentioned above, we propose a decomposition of the space of NTK eigenvectors into two orthogonal spaces: the nullspace $\mathcal{V}^2(t) := \mathcal{N}(t)$ and the space $\mathcal{V}^1(t) := \mathcal{N}^\perp(t)$ of eigenvectors associated with positive eigenvalues.

In the context of overparameterized NNs, the works [5], [8], [9] perform a similar decomposition, but the eigenvalues associated to the NTK matrix are always positive. Their decomposition is made between a space associated with high-value positive eigenvalues, and another space associated with low-value positive eigenvalues. Both ranges of values are separated by a "cutoff" value (a notion which is not necessary in our work). In these works, the training error converges to 0 (and not to a positive value equal to $\|\boldsymbol{v}^2(t_0)\|$ as in our work).

Furthermore, the theory of Rademacher complexity has been used in connection with rules of early stopping in the context of "kernel boosting" algorithms in [10], and in the context

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under grant agreement No. 101002857.

Daniel Martin Xavier, and Ludovic Chamoin are with Université Paris-Saclay, CentraleSupélec, ENS Paris-Saclay, CNRS, Laboratoire de Mécanique Paris-Saclay (LMPS), 91190, Gif-sur-Yvette, France (e-mail: daniel.martin-xavier@ens-paris-saclay.fr; ludovic.chamoin@ens-paris-saclay.fr)

Laurent Fribourg is with Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire de Méthodes Formelles (LMF), 91190, Gif-sur-Yvette, France. (e-mail: fribourg@lmf.cnrs.fr)

of overparameterized NNs in [11]–[13]. The results of these works are adapted here in the context of underparameterized NNs.

Few studies are dedicated to the analysis of underparameterized NNs: Wang et al. [14] used the notion of "activation patterns" to show that a loss of exponential type converges to 0 when the data is "well-separated". Besides, Bowman et al. [15] showed that the NN learns eigenfunctions of an integral operator determined by the NTK at rates corresponding to the eigenvalues ("spectral bias").

II. Preliminaries

A. Notation

In this paper, we denote by $\mathbb{R}$ and $\mathbb{N}$ the set of real and natural numbers, respectively. These symbols are annotated with subscripts to restrict them in the usual way, e.g., $\mathbb{R}_{>0}$ denotes the positive real numbers. We also denote by $\mathbb{R}^n$ an n-dimensional Euclidean space, and by $\mathbb{R}^{n\times m}$ a space of real matrices with n rows and m columns.

We use bold letters for vectors and bold capital letters for matrices. Given a matrix $\boldsymbol{A}$, let $\boldsymbol{A}_{i,j}$ be its (i,j)-th entry, $\lambda_{\min}(\boldsymbol{A})$ its minimal eigenvalue, and $\boldsymbol{A}^\top$ its transpose. The Euclidean norm is denoted by $\|\cdot\|$, the Frobenius norm by $\|\cdot\|_F$, and the inner product by $\langle\cdot,\cdot\rangle$. Let $\boldsymbol{I}_n$ be the $n\times n$ identity matrix, and $\sigma(\cdot)$ the ReLU function $\sigma(z)=\max\{z,0\}$.

We denote by $\mathbb{I}\{E\}$ the indicator function for an event E, by $B\uplus C$ the disjoint union of sets B and C, and by $\mathcal{V}^1\oplus\mathcal{V}^2$ the direct sum of vector spaces $\mathcal{V}^1$ and $\mathcal{V}^2$. We also use the abbreviation i.i.d. to indicate that a collection of random variables is independent and identically distributed. Finally, the time-discrete version of a time-continuous object ξ is denoted as $\tilde{\xi}$.

B. One-hidden Layer Neural Networks

We consider a one-hidden layer neural network with m neurons in the hidden layer and a ReLU activation function:

$$f(\boldsymbol{W},\boldsymbol{a},\boldsymbol{x})=\frac{1}{\sqrt{m}}\sum_{r=1}^{m}a_r\sigma(\boldsymbol{w}_r^\top\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x}\in\mathbb{R}^d$ is the input, $\boldsymbol{w}_1,\dots,\boldsymbol{w}_m\in\mathbb{R}^d$ are the weight vectors of the first layer, $a_1,\dots,a_m\in\mathbb{R}$ are the weights of the second layer. For simplicity, we denote $\boldsymbol{W}=(\boldsymbol{w}_1,\dots,\boldsymbol{w}_m)\in\mathbb{R}^{d\times m}$ and $\boldsymbol{a}=(a_1,\dots,a_m)^\top\in\mathbb{R}^m$.

In this work, we focus on the empirical risk minimization problem using a training dataset with n samples $S=\{(\boldsymbol{x}_i,y_i)\}_{i=1}^n$ drawn i.i.d. from an underlying data distribution $\mathcal{D}$ over $\mathbb{R}^d\times\mathbb{R}$. The NN is trained using a gradient descent (GD) algorithm on the following *training loss* function:

$$\mathcal{L}(\boldsymbol{W},\boldsymbol{a})=\frac{1}{2}\sum_{i=1}^{n}(y_i-f(\boldsymbol{W},\boldsymbol{a},\boldsymbol{x}))^2. \tag{2}$$

The parameters of the first layer are randomly initialized using $\boldsymbol{w}_r(0)\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$, while the parameters of the second layer are uniformly sorted using $a_r\sim\text{unif}(\{-1,1\}),\forall r\in[m]$. Similarly to other works on NTK [2], [5] (Section 3.1), we fix the parameters of the second layer of the NN and we only apply the GD algorithm over the weights of the first layer.

As the ReLU activation function used in the hidden layer is not continuously differentiable at 0, we should *a priori* consider the subgradient of $\sigma(0)$ which can take any value in $[0,1]$. However, Theorems 1 and 2 of [16] assert formally that the choice of $\sigma'(0)=s$ with s arbitrarily chosen in $[0,1]$ "does not affect" neither neural network training nor backpropagation. This justifies the use of $\sigma'(0)=0$, as in PyTorch [17].

The gradient formula is then given by:

$$\frac{\partial\mathcal{L}(\boldsymbol{W},\boldsymbol{a})}{\partial\boldsymbol{w}_r}=\frac{a_r}{\sqrt{m}}\sum_{i=1}^{n}(y_i-f(\boldsymbol{W},\boldsymbol{a},\boldsymbol{x}))\mathbb{I}\{\boldsymbol{w}_r^\top\boldsymbol{x}_i\geq 0\}\boldsymbol{x}_i. \tag{3}$$

The update rule is:

$$\tilde{\boldsymbol{W}}_{k+1}=\tilde{\boldsymbol{W}}_k-\eta\frac{\partial\mathcal{L}(\tilde{\boldsymbol{W}}k,\boldsymbol{a})}{\partial\boldsymbol{W}}, \tag{4}$$

where η is the learning rate and $\tilde{\boldsymbol{W}}_0=\boldsymbol{W}(0)$. This update rule corresponds to the Euler discretization of the set of ordinary differential equations defined by:

$$\frac{d\boldsymbol{W}(t)}{dt}=-\frac{\partial\mathcal{L}(\boldsymbol{W}(t),\boldsymbol{a})}{\partial\boldsymbol{W}}. \tag{5}$$

C. Gradient Descent for Neural Networks

According to Du et al. [2], the convergence of the GD algorithm to a globally optimal solution comes down to showing the convergence of the error (i.e. difference between the prediction of the NN and the ground truth) to zero. Their proof is based on the analysis of the error dynamics $\boldsymbol{v}:\mathbb{R}_{\geq 0}\to\mathbb{R}^n$ defined by:

$$\boldsymbol{v}(t)=\boldsymbol{p}(t)-\boldsymbol{y}, \tag{6}$$

where $\boldsymbol{p}(t)=(p_1(t),\dots,p_n(t))^\top\in\mathbb{R}^n$ is a vector with all n predictions $p_i(t)=f(\boldsymbol{W}(t),\boldsymbol{a},\boldsymbol{x}_i)$ at time t, and $\boldsymbol{y}=(y_1,\dots,y_n)^\top\in\mathbb{R}^n$.

As demonstrated in the proof of Theorem 3.2 by Du et al. [2], the continuous dynamics of the error $\boldsymbol{v}$ can be written in a compact way:

$$\frac{d}{dt}\boldsymbol{v}(t)=-\boldsymbol{H}[\boldsymbol{W}(t)]\boldsymbol{v}(t),\quad \boldsymbol{v}(0)=\boldsymbol{v}_0, \tag{7}$$

where $\boldsymbol{H}[\boldsymbol{W}(t)]:\mathbb{R}_{\geq 0}\to\mathbb{R}^{n\times n}$ is the Gram matrix from a kernel associated with the ReLU function, being symmetric positive semi-definite as follows:

$$\begin{aligned}\boldsymbol{H}_{ij}[\boldsymbol{W}(t)]&=\left\langle\frac{\partial f(\boldsymbol{W}(t),\boldsymbol{a},\boldsymbol{x}_i)}{\partial\boldsymbol{W}},\frac{\partial f(\boldsymbol{W}(t),\boldsymbol{a},\boldsymbol{x}_j)}{\partial\boldsymbol{W}}\right\rangle\\&=\frac{1}{m}\boldsymbol{x}_i^\top\boldsymbol{x}_j\sum_{r=1}^{m}\mathbb{I}\{\boldsymbol{x}_i^\top\boldsymbol{w}_r(t)\geq 0,\boldsymbol{x}_j^\top\boldsymbol{w}_r(t)\geq 0\}.\end{aligned}$$

The discrete version resulting from the Euler discretization of (7) reads:

$$\tilde{\boldsymbol{v}}_{k+1}-\tilde{\boldsymbol{v}}_k=-\eta\boldsymbol{H}[\tilde{\boldsymbol{W}}_k]\tilde{\boldsymbol{v}}_k, \tag{8}$$

where η is the step size, and $\tilde{\boldsymbol{v}}(0)=\tilde{\boldsymbol{v}}_0$.

We are now ready to formalize the problem:

Problem 1: Given the discrete time system in (8), provide conditions on the matrix $\boldsymbol{H}[\boldsymbol{W}(t)] : \mathbb{R}_{\geq 0} \to \mathbb{R}^{n\times n}$, the training loss function $\mathcal{L} : \mathbb{R}^{d\times m} \to \mathbb{R}_{\geq 0}$, and the step size η to ensure the convergence of the training error $\tilde{\boldsymbol{v}}_k$ to zero, together with an explicit bound on its convergence rate.

Let us recall the assumptions provided by Jerray et al. [4] to derive a solution for Problem 1:

Assumption C1: The GD algorithm for the update of the weights of the neural network in (4) converges to a local minimum $\boldsymbol{W}^*$, i.e. $\frac{\partial \mathcal{L}(\tilde{\boldsymbol{W}}_k, \boldsymbol{a})}{\partial \boldsymbol{W}}$ converges to 0 as k goes to infinity.

Assumption C2: There exist $\lambda^* > 0$ and $t_0 \geq 0$ such that, for all $t \geq t_0$: $\lambda_{\min}(\boldsymbol{H}[\boldsymbol{W}(t)]) \geq \lambda^*$, where the time-varying matrix $\boldsymbol{H}[\boldsymbol{W}(t)]$ is given in (7).

Assumption C3: The loss function $\mathcal{L}$ is locally strongly convex around every local minimizer $\boldsymbol{W}^*$ of $\mathcal{L}$, thus for every local minimizer $\boldsymbol{W}^*$, there is a neighborhood around $\boldsymbol{W}^*$ on which $\mathcal{L}$ is strongly convex.

As mentioned in Jerray et al. [4], Assumption C1 is satisfied when the gradient descent algorithm reaches a local minimum. This condition is satisfied almost surely with a random initialization of the algorithm [18], when the step size η is chosen such that $\eta < \frac{1}{L}$ where L is the Lipschitz constant of the loss function $\mathcal{L}$.

Furthermore, Assumption C3 is satisfied when the loss function $\mathcal{L}$ is strongly convex in the neighborhood of each minimizer. This condition can be always verified by initializing the parameters so that they fall into the basin of the local strong convexity region. In the case of the ReLU activation function, Zhong et al. [19] demonstrated that it satisfies certain mathematical properties, leading to local strong convexity (see Properties 3.1, 3.2, 3.3, Section 3).

Jerray et al. [4] then used Assumptions C1 and C3 to derive:

Theorem 1: (Theorem 1 of [4]) Under assumptions C1 and C3, if the step size η satisfies $\eta < \frac{2}{L}$, where L is the Lipschitz constant of the loss function $\mathcal{L}$, then the sequence $\|\tilde{\boldsymbol{W}}_k - \boldsymbol{W}(k\eta)\|$, $k \in \mathbb{N}$, converges to 0, where $\tilde{\boldsymbol{W}}_k$ is defined by (4) and $\boldsymbol{W}(k\eta) = [\boldsymbol{w}_1(k\eta), \ldots, \boldsymbol{w}_m(k\eta)]$ with $\boldsymbol{W}(t)$ defined by (5).

Since $\|\tilde{\boldsymbol{W}}_k - \boldsymbol{W}(k\eta)\|$ converges to 0 according to Theorem 1, it follows by continuity that for all $i \in \{0, \ldots, n-1\}$:

$$\lim_{k\to\infty} \|\tilde{\lambda}_i(k\eta) - \lambda_i(k\eta)\| = 0, \tag{9}$$

where $\tilde{\lambda}_i$ (resp. λ_i) is the i-th eigenvalue of $\boldsymbol{H}[\tilde{\boldsymbol{W}}_k]$ (resp. $\boldsymbol{H}[\boldsymbol{W}(t)]$).

On the other hand, the authors used Assumption C2 to derive the following statement:

Theorem 2: (Theorem 2 of [4]) Under assumptions C1, C2 and C3, if the step size η satisfies $\eta < \frac{2}{L}$, where L is the Lipschitz constant of the training loss function $\mathcal{L}$, then there exists $k_0 \in \mathbb{N}$ such that, for all $k \geq k_0$:

$$\|\tilde{\boldsymbol{v}}_k\| \leq (1 - \frac{1}{2}\lambda^*\eta)^{k-k_0}\|\tilde{\boldsymbol{v}}_{k_0}\|.$$

Assumption C2 only holds in the case of overparameterized NNs, in which all eigenvalues of the NTK are larger than some $\lambda^* > 0$ (see [4]). The present work considers the more general case where the eigenvalues $\lambda_i(t)$ of $\boldsymbol{H}[\boldsymbol{W}(t)]$ are divided into: one part always larger than λ^*, and another part always equal to zero (corresponding to the nullspace $\mathcal{N}$).

III. Training Error

In underparameterized networks, Assumption C2 does not hold and, without loss of generality, we suppose that the space $\mathcal{V}(t)$ of eigenvectors of $\boldsymbol{H}[\boldsymbol{W}(t)]$ decomposes as $\mathcal{V}^1(t) \oplus \mathcal{V}^2(t)$, where $\mathcal{V}^1(t)$ (resp. $\mathcal{V}^2(t)$) is the cluster of eigenvectors associated to eigenvalues $\lambda(t)$ greater than 0 (resp. equal to 0). Formally, we replace C2 by:

Assumption C2': There exist $\lambda^* \in \mathbb{R}_{>0}$, $I_1 = \{0, \ldots, n_1 - 1\}$ and $I_2 = \{n_1, \ldots, n_1 + n_2 - 1\}$ with $n_1 + n_2 = n$ such that the eigenvalues can be divided into:

- $\lambda_i(t) \geq \lambda^*$ for all $i \in I_1, t \geq 0$,
- $\lambda_i(t) = 0$ for all $i \in I_2, t \geq 0$.

where $\{\lambda_i(t)\}_{i\in I_1 \uplus I_2}$ is the eigenvalues set of the NTK matrix $\boldsymbol{H}[\boldsymbol{W}(t)]$, and I_1 (resp. I_2) the index set of eigenvectors spanning $\mathcal{V}^1(t)$ (resp. $\mathcal{V}^2(t)$).

If we assume C2' instead of C2 , the result of Theorem 2 becomes:

Theorem 3: Under assumptions C1, C2' and C3, there exists $\alpha \in (0, \lambda^*)$ such that, if the step size η satisfies $\eta < \frac{2}{L}$ and $\eta < \frac{1}{\alpha}$, where L is the Lipschitz constant of the loss function $\mathcal{L}$, then there exists k_0 such that for all $k \geq k_0$:

$$\|\tilde{\boldsymbol{v}}_k\| \leq \sqrt{(1-\alpha\eta)^{2(k-k_0)}\|\tilde{\boldsymbol{v}}^1_{k_0}\|^2 + \|\tilde{\boldsymbol{v}}^2_{k_0}\|^2}, \tag{10}$$

where $\tilde{\boldsymbol{v}}^i_k$ $(i = 1, 2)$ is the projection of the vector $\tilde{\boldsymbol{v}}_k$ on the span $\tilde{\mathcal{V}}^i_k$ of the n_i eigenvectors of $\boldsymbol{H}[\tilde{\boldsymbol{W}}_k]$. It follows that $\limsup_{k\to\infty} \|\tilde{\boldsymbol{v}}_k\| \leq b := \|\tilde{\boldsymbol{v}}^2_{k_0}\|$.

Proof: By Theorem 1, which relies on C1 and C3, but not C2, we know that for $\eta < \frac{2}{L}$, $\|\tilde{\boldsymbol{W}}_k - \boldsymbol{W}(k\eta)\|$ converges to 0 as $k \to \infty$. Let $\tilde{\lambda}_i(k\eta)$ $(i \in \{0, \ldots, n-1\})$ denote the eigenvalues of the matrix $\boldsymbol{H}[\tilde{\boldsymbol{W}}_k]$. Recall that C2' specifies that the eigenvalues $\lambda_i(t)$ of $\boldsymbol{H}[\boldsymbol{W}(t)]$ with $i \in I_1$ are in $[\lambda^*, \infty)$, and those with $i \in I_2$ are null. It follows from (9), that there exists $\alpha \in (0, \lambda^*)$ such that:

$$\lim_{k\to\infty} \tilde{\lambda}_i(k\eta) > \alpha \text{ for all } i \in I_1, \tag{11}$$

$$\lim_{k\to\infty} \tilde{\lambda}_i(k\eta) = 0 \text{ for all } i \in I_2. \tag{12}$$

So there exists k_0 such that for all $k \geq k_0$:

$$\tilde{\lambda}_i(k\eta) \geq \alpha \text{ for all } i \in I_1, \tag{13}$$

$$\tilde{\lambda}_i(k\eta) < \alpha \text{ for all } i \in I_2. \tag{14}$$

It follows that for all $k \geq k_0$, $\tilde{\mathcal{V}}^1_k$ and $\tilde{\mathcal{V}}^2_k$ are orthogonal (since each eigenvalue $\tilde{\lambda}_i(k\eta)$ with $i \in I_1$ is different from each eigenvalue $\tilde{\lambda}_j(k\eta)$ with $j \in I_2$). The discretized dynamics of the error in (8) then writes:

$$\tilde{\boldsymbol{v}}_{k+1} = (\boldsymbol{I}_n - \eta\boldsymbol{H}[\tilde{\boldsymbol{W}}_k])\tilde{\boldsymbol{v}}_k. \tag{15}$$

We also know that the NTK matrix $\boldsymbol{H}[\tilde{\boldsymbol{W}}_k]$ can be decomposed as:

$$\boldsymbol{H}[\tilde{\boldsymbol{W}}_k] = \boldsymbol{P}_k\boldsymbol{D}_k\boldsymbol{P}_k^\top, \tag{16}$$

where $\boldsymbol{P}_k$ is the transition matrix whose columns are the eigenvectors of $\boldsymbol{H}[\tilde{\boldsymbol{W}}_k]$, and $\boldsymbol{D}_k$ is the diagonal eigenvalue

matrix with first n_1 elements in $[\alpha, \infty)$, and last n_2 elements in $[0, \alpha)$. From (15), it follows:

$$\tilde{\boldsymbol{v}}_{k+1} = \boldsymbol{P}_k(\boldsymbol{I}_n - \eta \boldsymbol{D}_k)\boldsymbol{P}_k^\top \tilde{\boldsymbol{v}}_k. \tag{17}$$

The $n \times n$ matrix $\boldsymbol{P}_k$ is made of an upper $n_1 \times n$ matrix $\boldsymbol{P}_k^1$ and a lower $n_2 \times n$ matrix $\boldsymbol{P}_k^2$. The rows of the matrix $\boldsymbol{P}_k^1$ are the n_1 eigenvectors of $\boldsymbol{H}[\tilde{\boldsymbol{W}}_k]$ associated with eigenvalues in $[\alpha, \infty)$, and those of $\boldsymbol{P}_k^2$ are the n_2 other eigenvectors. Hence, for a vector $\boldsymbol{v}$, ${\boldsymbol{P}_k^i}^\top \boldsymbol{v}$ $(i = 1, 2)$ corresponds to the projection $\tilde{\boldsymbol{v}}_k^i$ of $\boldsymbol{v}$ on the span $\tilde{\mathcal{V}}_k^i$ of the n_i eigenvectors of $\boldsymbol{H}[\tilde{\boldsymbol{W}}_k]$. Equation (17) thus decomposes as:

$$\tilde{\boldsymbol{v}}_{k+1}^1 = \boldsymbol{P}_k^1(\boldsymbol{I}_{n_1} - \eta \boldsymbol{D}_k^1){\boldsymbol{P}_k^1}^\top \tilde{\boldsymbol{v}}_k^1, \tag{18}$$

$$\tilde{\boldsymbol{v}}_{k+1}^2 = \boldsymbol{P}_k^2(\boldsymbol{I}_{n_2} - \eta \boldsymbol{D}_k^2){\boldsymbol{P}_k^2}^\top \tilde{\boldsymbol{v}}_k^2, \tag{19}$$

where $\boldsymbol{D}_k^1$ is the $n_1 \times n_1$ top left submatrix of $\boldsymbol{D}_k$, and $\boldsymbol{D}_k^2$ the $n_2 \times n_2$ bottom right submatrix. For $k \geq k_0$ and $\eta < \frac{1}{\alpha}$, equations (18) and (19) lead to:

$$\|\tilde{\boldsymbol{v}}_{k+1}^1\| \leq \|\boldsymbol{I}_{n_1} - \eta \boldsymbol{D}_k^1\| \|\tilde{\boldsymbol{v}}_k^1\| \leq (1 - \alpha\eta)\|\tilde{\boldsymbol{v}}_k^1\|, \tag{20}$$

$$\|\tilde{\boldsymbol{v}}_{k+1}^2\| \leq \|\boldsymbol{I}_{n_2} - \eta \boldsymbol{D}_k^2\| \|\tilde{\boldsymbol{v}}_k^2\| \leq \|\tilde{\boldsymbol{v}}_k^2\|. \tag{21}$$

It follows from (20) and (21) that there exists k_0 such that for all $k \geq k_0$ and $\eta < \frac{1}{\alpha}$:

$$\begin{aligned} \|\tilde{\boldsymbol{v}}_k^1\| &\leq (1 - \alpha\eta)^{k-k_0} \|\tilde{\boldsymbol{v}}_{k_0}^1\|, \\ \|\tilde{\boldsymbol{v}}_k^2\| &\leq \|\tilde{\boldsymbol{v}}_{k_0}^2\|. \end{aligned}$$

Finally, using the fact that $\tilde{\mathcal{V}}_k^1$ and $\tilde{\mathcal{V}}_k^2$ are orthogonal:

$$\begin{aligned} \|\tilde{\boldsymbol{v}}_k\|^2 &= \|\tilde{\boldsymbol{v}}_k^1\|^2 + \|\tilde{\boldsymbol{v}}_k^2\|^2, \\ &\leq (1 - \alpha\eta)^{2(k-k_0)} \|\tilde{\boldsymbol{v}}_{k_0}^1\|^2 + \|\tilde{\boldsymbol{v}}_{k_0}^2\|^2. \end{aligned} \tag{22}$$

■

Remark 1: In practice k_0 is small, which means that $\|\tilde{\boldsymbol{v}}_k^2\|$ is rapidly constant from $t_0 = k_0\eta$. The training error $\|\tilde{\boldsymbol{v}}_k\|$ is thus asymptotically equal to $b = \|\tilde{\boldsymbol{v}}_{k_0}^2\|$. The constant b depends on the initial value $\boldsymbol{W}(0)$ of the NN parameters. When $b = \|\tilde{\boldsymbol{v}}_{k_0}^2\|$ is deemed too important, one can stop the GD procedure early (at $t = k_0\eta$) and reinitialize it. Note that, apart from the choice of initial weight $\boldsymbol{W}(0)$, the GD procedure considered here is deterministic.

Example 1: In order to verify that the training error converges linearly to a non-zero constant on underparametrized NNs, we consider a toy example using the Van der Pol oscillator (see e.g. [20]). The system possesses 2 states $\boldsymbol{x} = (x_1, x_2)$, a control action u, and a damping coefficient $\mu = 1$. The oscillator position is represented by x_1, its velocity by x_2, and the state derivative by $\dot{\boldsymbol{x}}$. The system is defined by:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \mu(1 - x_1^2)x_2 - x_1 + u. \end{cases} \tag{23}$$

The goal is to design a NN that mimics the behavior of an optimization problem in a MPC controller that steers the system to a desired position x_1^{ref}. We first collect data using an implementation of the MPC controller, which is then used to train a one-hidden layer neural network offline.

The MPC controller is synthesized using a time step of $T_s = 0.5$s, a prediction horizon of $N = 5$, and an initial condition $\boldsymbol{x}_0 = (1, 0)$ using [21]. The command applied to the oscillator is constrained to the interval $u \in [-1, 1]$, and the cost function associated to the optimization is written as:

$$\begin{aligned} J(\boldsymbol{x}, u) = \sum_{i=0}^{N-1} &\|x_1^{ref}(k+i) - x_1(k+i)\| + \gamma \|\Delta u(k+i)\| \\ \text{s.t.} \quad & u_{lb} \leq u(k+i) \leq u_{ub}, \quad \forall i \in [0, ..., N-1] \end{aligned}$$

where u_{lb} and u_{ub} are the lower and upper bounds for the command, respectively, $\Delta u(k) = u(k) - u(k-1)$ is the change on the command, and $\gamma = 0.1$ is a weighting coefficient. The error with respect to the reference is represented by $\epsilon(k) = x_1^{ref}(k) - x_1(k)$, where k is the current step time. The MPC simulation is conducted offline through the definition of different setpoints that are randomly generated in $[-1, 1]$.

The NN used to simulate the MPC is a one-step-ahead predictive controller depicted in Fig. 1. It has $d = 3$ entries in the input layer $(x_1(k), x_2(k), x_1^{ref}(k))$, 10 neurons in the hidden layer, and one output $(u(k))$ in the last layer. It was implemented using PyTorch [17] with a ReLU activation function in the hidden layer.

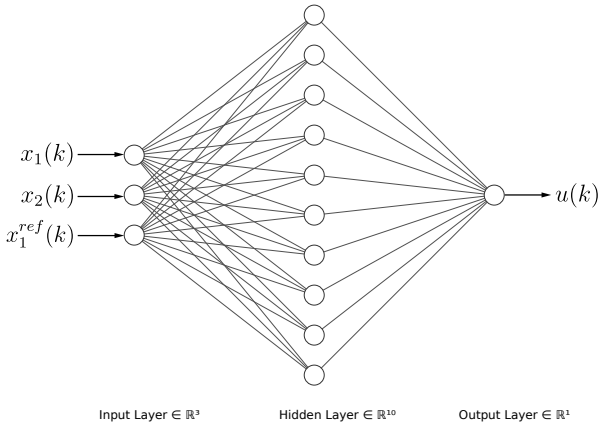

Fig. 1. Architecture of the NN used to simulate the MPC controller of Example 1.

The GD algorithm is performed over a single batch of $n = 15$ data. The training phase is conducted over $2 \cdot 10^5$ epochs with a learning rate[1] of $\eta = 10^{-2}$. Under this configuration, the 15×15 NTK matrix $\boldsymbol{H}[\tilde{\boldsymbol{W}}_k]$ describes the dynamics of the error $\|\tilde{\boldsymbol{v}}_k\|$ at each iteration k. Figure 2 depicts the log-scale evolution of the subset of eigenvalues $\tilde{\lambda}_i$ with $i \in I_1 = \{0, \ldots, 8\}$ which are larger than $\alpha = 10^{-4}$. The other eigenvalues $\tilde{\lambda}_i$ with $i \in I_2 = \{9, \ldots, 14\}$ are inferior to $\alpha = 10^{-4}$ and correspond to the eigenvectors spanning $\mathcal{N}$.

Furthermore, the experimental proof is illustrated in Fig. 3, which depicts the log-scale evolution of the error $\|\tilde{\boldsymbol{v}}_k\|$ (blue), and projections $\|\tilde{\boldsymbol{v}}_k^1\|$ (orange) and $\|\tilde{\boldsymbol{v}}_k^2\|$ (green). The error $\|\tilde{\boldsymbol{v}}_k^1\|$ converges linearly to 0, and we have: $\limsup_{k\to\infty} \|\tilde{\boldsymbol{v}}_k\| \leq b := \|\tilde{\boldsymbol{v}}_{k_0}^2\| = 0.0584$ for $k_0 = 17,000$. The red curve corresponds to the right-hand side of (10), and is above $\|\tilde{\boldsymbol{v}}_k\|$ (blue) as stated by Theorem 3.

[1]Note that according to Theorem 3, $\eta < 2/L$ and $\eta < 1/\alpha$. In practice, the step size η is determined by decreasing it sufficiently since α separates the positive eigenvalues from the zero eigenvalues in the continuous setting.

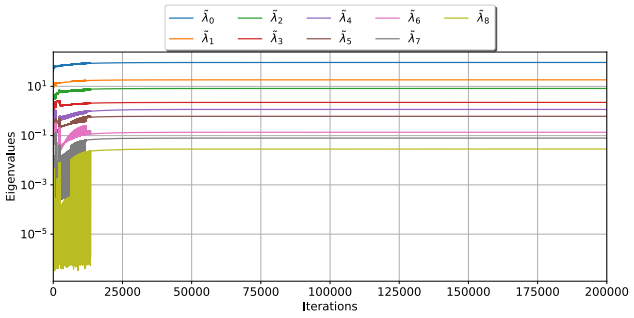

Fig. 2. Evolution of the 9 positive eigenvalues $(\tilde{\lambda}_0, \ldots, \tilde{\lambda}_8)$ of the NTK matrix $\boldsymbol{H}[\tilde{\boldsymbol{W}}_k]$ (the 6 other eigenvalues are inferior to $\alpha = 10^{-4}$ and are not shown in the figure).

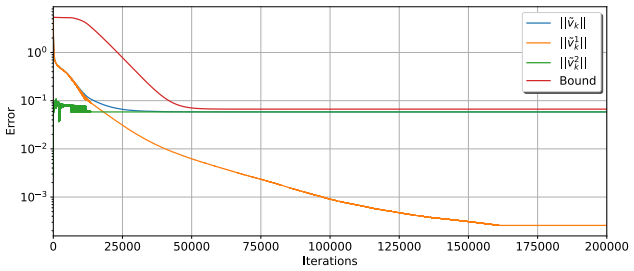

Fig. 3. Log-scale evolution of the training error $\|\tilde{\boldsymbol{v}}_k\|$ (blue), which is decomposed in $\|\tilde{\boldsymbol{v}}_k^1\|$ (orange) and $\|\tilde{\boldsymbol{v}}_k^2\|$ (green). The red curve corresponds to the right-hand side of (10), and is above $\|\tilde{\boldsymbol{v}}_k\|$ (blue) as stated by Theorem 3.

IV. Generalization Error

In this section, we give an upper bound Γ for the *generalization error* (see Proposition 1) which follows from results of Arora et al. [5]. Let $\ell(\cdot,\cdot)$ be an elementary loss function defined over $\mathbb{R} \times \mathbb{R}$. The *population loss* $L_{\mathcal{D}}$ over data distribution $\mathcal{D}$ and the *empirical loss* $L_{\mathcal{S}}$ over n samples $S = \{(x_i, y_i)\}_{i=1}^n$ drawn i.i.d. from $\mathcal{D}$ are defined as follows (see [5]):

$$L_{\mathcal{D}}(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\ell(f(\boldsymbol{x}), y], \tag{24}$$

$$L_S(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), y_i). \tag{25}$$

The *generalization error* refers to $L_{\mathcal{D}}(f) - L_S(f)$ for the learned function f given sample S. Given a class $\mathcal{F}$ of functions, the notion of *Rademacher complexity* (denoted $\mathcal{R}_S(\mathcal{F})$) has been used in [22] to derive an upper bound for the generalization error:

Theorem 4: (Theorem 11.3 of [22]) Suppose the loss function $\ell(\cdot,\cdot)$ is bounded in $[0, c]$ and is ρ-Lipschitz in the first argument. Then with probability at least $1-\delta$ over sample S of size n:

$$\sup_{f\in\mathcal{F}}\{L_{\mathcal{D}}(f) - L_S(f)\} \leq 2\rho\mathcal{R}_S(\mathcal{F}) + 3c\sqrt{\frac{\log(2/\delta)}{2n}},$$

where the Rademacher complexity is defined as:

$$\mathcal{R}_S(\mathcal{F}) := \frac{1}{n}\mathbb{E}_{\varepsilon\in\{\pm1\}^n}\left[\sup_{f\in\mathcal{F}}\sum_{i=1}^{n}\varepsilon_i f(\mathbf{x}_i)\right]. \tag{26}$$

Besides, Arora et al. [5] proved the following:

Theorem 5: (Lemma 5.4 of [5]) Given $R > 0$, with probability at least $1-\delta$ over the random initialization $(\boldsymbol{W}(0), \boldsymbol{a})$, simultaneously for every $B > 0$, the following function class:

$$\mathcal{F}_{R,B}^{\boldsymbol{W}(0),\boldsymbol{a}} = \{f(\boldsymbol{W},\boldsymbol{a}) : \|\boldsymbol{w}_r - \boldsymbol{w}_r(0)\| \leq R\ (\forall r\in[m]),$$
$$\|\boldsymbol{W} - \boldsymbol{W}(0)\|_F \leq B\} \tag{27}$$

has empirical Rademacher complexity:

$$\mathcal{R}_S(\mathcal{F}_{R,B}^{\boldsymbol{W}(0),\boldsymbol{a}}) := \frac{1}{n}\mathbb{E}_{\varepsilon\in\{\pm1\}^n}\left[\sup_{f\in\mathcal{F}_{R,B}^{\boldsymbol{W}(0),\boldsymbol{a}}}\sum_{i=1}^{n}\varepsilon_i f(\mathbf{x}_i)\right]$$

bounded as:

$$\mathcal{R}_S(\mathcal{F}_{R,B}^{\boldsymbol{W}(0),\boldsymbol{a}}) \leq \frac{B}{\sqrt{2n}}\left(1 + \left(\frac{2\log\frac{2}{\delta}}{m}\right)^{1/4}\right)$$
$$+ 2R^2\sqrt{m} + R\sqrt{2\log\frac{2}{\delta}}.$$

It follows from Theorem 4 and Theorem 5:

Proposition 1: Consider the function loss $\ell(\cdot,\cdot)$ defined by $\ell(z,y) := (z-y)^2$ for all $z, y \in \mathbb{R}$, and suppose $\ell(\cdot,\cdot)$ is bounded in $[0,c]$ and is ρ-Lipschitz in its first argument. Then with probability at least $1-\delta$ over sample S:

$$\sup_{f\in\mathcal{F}_{R,B}^{\boldsymbol{W}(0),\boldsymbol{a}}}\{L_{\mathcal{D}}(f) - L_S(f)\} \leq \Gamma$$

with

$$\Gamma = 2\rho R\left(2R\sqrt{m} + \sqrt{2\log\frac{2}{\delta}}\right)$$
$$+\frac{1}{\sqrt{2n}}\left(2\rho B\left[1 + \left(\frac{2\log\frac{2}{\delta}}{m}\right)^{1/4}\right] + 3c\sqrt{\log(2/\delta)}\right).$$

Given a test set $S_{test} = \{(\boldsymbol{x}_i^{test}, y_i^{test})\}_{i=1}^{n_{test}}$ of the NN, the *test error*, i.e. the mean squared error associated to S_{test}, is defined by:

$$L_{test} = \frac{1}{n_{test}}\sum_{i=1}^{n_{test}}(f_{\boldsymbol{W}^*,\boldsymbol{a}}(\boldsymbol{x}_i^{test}) - y_i^{test})^2, \tag{28}$$

where $f_{\boldsymbol{W}^*,\boldsymbol{a}}$ is the learned function from the GD algorithm on the training set S. Knowing that $L_{test} \leq L_{\mathcal{D}}$, we can use Proposition 1 to write:

$$L_{test} \leq \Sigma, \tag{29}$$

where $\Sigma := \Gamma + L_S$.

Example 2: We consider the same case study as in Example 1, except that we increase the number of neurons in the hidden layer to $m = 100$ and we vary the number of data ($8 \leq n \leq 400$). The network is trained using GD until convergence with the same hyperparameters and initialization as in Example 1.

Figure 4 gives the evolution of L_{test} and Σ (bound over the population loss $L_{\mathcal{D}}$) as a function of the number of training data n. We verify that the curves L_{test} and Σ have similar trends.[2] In particular, the curves present a double-peak shape

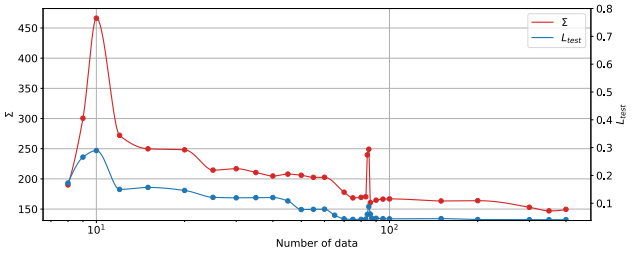

Fig. 4. Evolution of L_{test} and Σ with the number of training data n.

as observed, e.g., in [6] and [7] with peaks around $n = d$ and $n = m$.

Finally, Fig. 5 depicts a comparison between the command u^{MPC} synthesized by the MPC controller and u^{NN} predicted by the neural network (with $m = 100$ neurons and a training set of size $n = 2160$). This illustrates the fact that an underparameterized network is capable of reproducing the MPC controller with good accuracy ($L_{test} = 0.003$).

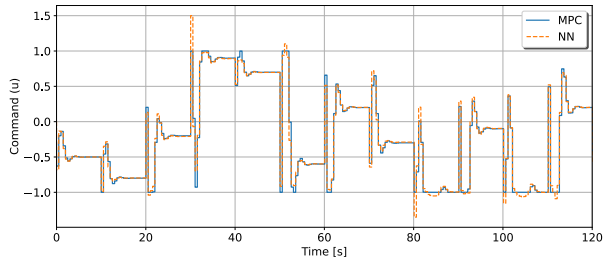

Fig. 5. Comparison of the true command (u^{MPC}) and the one predicted by the NN (u^{NN}) using the test dataset.

V. Conclusion

This paper presented a proof that the training loss of an underparameterized neural network converges linearly to a (non-null) constant. The generalization error of the trained model was investigated, and a bound to the population loss was computed using the Rademacher complexity. We illustrate these results on a one-hidden layer NN simulating a MPC controller. We hope that our work will pave the way for a better understanding of the success of data-driven MPC using neural networks of moderate size.

Limitations and Future Work

This study focused on the analysis of the training error convergence in a neural network with scalar output where only the first layer is trained. We plan to extend the analysis to include training on several layers and to consider multidimensional outputs.

[2]Note that, in this example, the difference between Σ and L_{test} is large. This is partly explained by the fact that L_{test} has been computed here with a set S_{test} of data obtained with high precision. A set S_{test} of data perturbed with noise would increase the value of L_{test} and bring it closer to Σ.

References

[1] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *CoRR*, vol. abs/1806.07572, 2018.

[2] S. S. Du, X. Zhai, B. Póczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," *CoRR*, vol. abs/1810.02054, 2018.

[3] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," *CoRR*, vol. abs/1811.03804, 2019.

[4] J. Jerray, A. Saoud, and L. Fribourg, "Using Euler's method to prove the convergence of neural networks," *IEEE Control. Syst. Lett.*, vol. 6, 2022.

[5] S. Arora, S. S. Du, W. Hu, Z. Li, and R. Wang, "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks," *CoRR*, vol. abs/1901.08584, 2019.

[6] S. d'Ascoli, L. Sagun, and G. Biroli, "Triple descent and the two kinds of overfitting: Where & why do they appear?" *CoRR*, vol. abs/2006.03509, 2020.

[7] T. J. Viering and M. Loog, "The shape of learning curves: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, 2023.

[8] S. Oymak and M. Soltanolkotabi, "Toward moderate overparameterization: Global convergence guarantees for training shallow neural networks," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, 2020.

[9] S. Oymak, Z. Fabian, M. Li, and M. Soltanolkotabi, "Generalization, adaptation and low-rank representation in neural networks," in *53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019.

[10] Y. Wei, F. Yang, and M. J. Wainwright, "Early stopping for kernel boosting algorithms: A general analysis with localized complexities," in *NeurIPS 2017*, 2017.

[11] M. Li, M. Soltanolkotabi, and S. Oymak, "Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks," in *The 23rd International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.

[12] N. Golowich, A. Rakhlin, and O. Shamir, "Size-independent sample complexity of neural networks," in *Conference On Learning Theory, COLT 2018*. PMLR, 2018.

[13] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," in *Annual Conference on Neural Information Processing Systems*, 2019.

[14] M. Wang and C. Ma, "Early stage convergence and global convergence of training mildly parameterized neural networks," in *NeurIPS*, 2022.

[15] B. Bowman and G. Montúfar, "Implicit bias of MSE gradient optimization in underparameterized neural networks," *CoRR*, vol. abs/2201.04738, 2022.

[16] D. Bertoin, J. Bolte, S. Gerchinovitz, and E. Pauwels, "Numerical influence of ReLU'(0) on backpropagation," *CoRR*, vol. abs/2106.12915, 2021.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.

[18] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht, "Gradient descent converges to minimizers," *CoRR*, vol. abs/1602.04915, 2016.

[19] K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon, "Recovery guarantees for one-hidden-layer neural networks," in *International Conference on Machine Learning*, 2017.

[20] E. A. Antonelo, E. Camponogara, L. O. Seman, E. R. de Souza, J. P. Jordanou, and J. F. Hubner, "Physics-informed neural nets for control of dynamical systems," *CoRR*, vol. abs/2104.02556, 2022.

[21] S. Lucia, A. Tatulea-Codrean, C. Schoppmeyer, and S. Engell, "Rapid development of modular and sustainable nonlinear model predictive control solutions," *Control Engineering Practice*, vol. 60, 2017.

[22] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. The MIT Press, 2012.