Towards a Mathematical Understanding of the Difficulty in Learning with Feedforward Neural Networks

Hao Shen
fortiss - The Research Institute of the Free State of Bavaria, Germany
Guerickestr. 25, 80805 Munich, Germany
hao.shen@fortiss.org

Abstract

Training deep neural networks for solving machine learning problems is one great challenge in the field, mainly due to its associated optimisation problem being highly non-convex. Recent developments have suggested that many training algorithms do not suffer from undesired local minima under certain scenario, and consequently led to great efforts in pursuing mathematical explanations for such observations. This work provides an alternative mathematical understanding of the challenge from a smooth optimisation perspective. By assuming exact learning of finite samples, sufficient conditions are identified via a critical point analysis to ensure any local minimum to be globally minimal as well. Furthermore, a state of the art algorithm, known as the Generalised Gauss-Newton (GGN) algorithm, is rigorously revisited as an approximate Newton's algorithm, which shares the property of being locally quadratically convergent to a global minimum under the condition of exact learning.

1. Introduction

Deep Neural Networks (DNNs) have been successfully applied to solve challenging problems in pattern recognition, computer vision, and speech recognition [3, 21, 43]. Despite this success, training DNNs is still one of the greatest challenges in the field [9]. In this work, we focus on training the classic feedforward Multi-Layer Perceptrons (MLPs). It is known that performance of MLPs is highly dependent on various factors in a very complicated way. For example, studies in [15, 37] identify the topology of MLPs as a determinative factor. Works in [25, 9] demonstrate the impact of different activation functions to performance of MLPs. Moreover, a choice of error/loss functions is also shown to be influential as in [8].

Even with a well designed MLP architecture, training a specific MLP both effectively and efficiently can be as challenging as constructing the network. The most popular method used in training MLPs is the well-known *backpropagation* (BP) algorithm [42]. Although the classic BP algorithm shares a great convenience of being very simple, they can suffer from two major problems, namely, (i) convergence to undesired local minima, if global optimality is assumed; and (ii) slow convergence speed. Early works as [38, 30] argue that such problems with BP algorithms are essentially due to their nature of being gradient descent algorithms, while an associated optimisation problem for MLP training is often highly non-convex and of large scale.

One major approach to address the problem of undesired local minima in MLP training is via an error/loss surface analysis [14, 35, 6]. Even for simple tasks, such as the classic XOR problem, the error surface analysis is surprisingly complicated and its results can be hard to conclude [22, 35, 36]. Early efforts in [11, 44, 45] try to identify general conditions on the topology of MLPs to eliminate undesired local minima, i.e., suboptimal local minima. Unfortunately, these attempts fail to provide complete solutions to general problems. On the other hand, although BP algorithms are often thought to be sensitive to initialisations [19], recent results reported in [10] suggest that modern MLP training algorithms can overcome the problem of suboptimal local minima conveniently. Such observations have triggered several very recent efforts to characterise global optimality of DNN training [17, 28, 13].

The work in [17] shows that both *deep linear networks* and *deep nonlinear networks* with only the Rectified Linear Unit (ReLU) function in the hidden layers are free of suboptimal local minima. The attempted technique is not applicable for analysing *deep nonlinear networks* with other activation functions, e.g. the *Sigmoid* and the *SoftSign*. Recent work in [28] proves that all local minima are globally minimal for exact learning with wide MLPs, if the number of units in a hidden layer of the network is larger than the number of training samples and the network structure from that layer on is pyramidal. Unfortunately, the deployed techniques can neither exclude the possibility of suboptimal local minima of low rank, nor be applied to narrow MLPs.

Most recently, by restricting the network output and its regularisation to be positively homogeneous with respect to the network parameters, the work in [13] develops sufficient conditions to guarantee all local minima to be globally minimal. So far, such results only apply to networks with either one hidden layer or multiple deep subnetworks connected in parallel. Moreover, it still fails to explain performance of MLPs with non-positively homogeneous activation functions.

To deal with slow convergence speed of the classic BP algorithm, various modifications have been developed, such as momentum based BP algorithm [40], conjugate gradient algorithm [5], BFGS algorithm [20], and ADAptive Moment estimation (ADAM) algorithm [18]. Specifically, a Generalised Gauss-Newton (GGN) algorithm proposed in [31] has demonstrated its prominent performance over many state of the art training algorithms in practice [23, 39]. Unfortunately, justification for such performance is still mathematically vague [24]. Another popular solution to deal with slow convergence is to employ Newton's method for MLP training. However, an implementation of exact Newton's method is often computationally prohibitive. Hence, approximations of the Hessian matrix are needed to address the issue, such as a diagonal approximation structure [1], and a block diagonal approximation structure [41]. However, without a complete evaluation of the true Hessian, performance of these heuristic approximations is hardly convincing. Although existing attempts [2, 27] have characterised the Hessian by applying partial derivatives, these results still fail to provide further structural information of the Hessian, due to the limitations of partial derivatives.

In this work, we provide an alternative mathematical understanding of the difficulty in training MLPs from a smooth optimisation perspective. Sufficient conditions are identified via a critical point analysis to ensure all local minima to be globally minimal. Convergence properties of the GGN algorithm are rigorously analysed as an approximate Newton's algorithm.

2. Learning with MLPs

Many machine learning tasks can be formulated as a problem of learning a task-specific *ground truth mapping* $g^*\colon \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$, where $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ denote an *input space* and an *output space*, respectively. The problem of interest is to approximate g^*, given only a finite number of samples in either $\boldsymbol{\mathcal{X}}$ or $\boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}}$. When only T samples in $\boldsymbol{\mathcal{X}}$ are given, say $\{x_i\}_{i=1}^T \subset \boldsymbol{\mathcal{X}}$, the problem of approximating g^* is known as *unsupervised learning*. When both input samples and their desired outputs $y_i := g^*(x_i)$ are provided, i.e., given training samples $\{(x_i, y_i)\}_{i=1}^T \subset \boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}}$, the corresponding problem is referred to as *supervised learning*. In this work, we only focus on the problem of supervised learning.

A popular approach to evaluate learning outcomes is via minimising an empirical error/loss function as

$$\widetilde{g} \in \operatorname*{argmin}_{g\in\boldsymbol{\mathcal{G}}} \frac{1}{T}\sum_{i=1}^{T} E\big(g(x_i), p(x_i)\big), \tag{1}$$

where $\boldsymbol{\mathcal{G}}$ denotes a hypothetical function space, where a minimiser $\widetilde{g}$ is assumed to be reachable, and $E\colon \boldsymbol{\mathcal{Y}}\times\boldsymbol{\mathcal{Z}} \to \mathbb{R}$ denotes a suitable error function that evaluates the estimate $g(x_i)$ against some task-dependent prior knowledge $p(x_i) \in \boldsymbol{\mathcal{Z}}$. For supervised learning problems, such prior knowledge is simply the corresponding desired output y_i, i.e., $p(x_i) := g^*(x_i)$. In general, given only a finite number of samples, the ground truth mapping g^* is hardly possible to be exactly learned as the solution $\widetilde{g}$. Nevertheless, we can still define a specific scenario of exact learning with respect to a finite number of samples.

Definition 1 (Exact learning)**.** *Let $g^*\colon \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ be the ground truth mapping. Given samples $\{x_i, y_i\}_{i=1}^T \subset \boldsymbol{\mathcal{X}}\times\boldsymbol{\mathcal{Y}}$, a mapping $\widehat{g}\colon \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$, which satisfies $\widehat{g}(x_i) = g^*(x_i)$ for all $i = 1, \ldots, T$, is called a* finite exact approximator *of g^* based on the T samples.*

For describing MLPs, we denote by L the number of layers in an MLP structure, and by n_l the number of processing units in the l-th layer with $l = 1, \ldots, L$. Specifically, by $l = 0$, we refer it to the input layer. Let $\sigma\colon \mathbb{R} \to \mathbb{R}$ be a unit activation function, which was traditionally chosen to be non-constant, bounded, continuous, and monotonically increasing. Recent choices, e.g. ReLU, SoftPlus, and Bent identity, are unbounded functions. In this work, we restrict activation functions to be *smooth* and *monotonically increasing*, and denote by $\sigma'\colon \mathbb{R} \to \mathbb{R}$ and $\sigma''\colon \mathbb{R} \to \mathbb{R}$ the first and second derivative of the activation function σ, respectively.

For the (l, k)-th unit in an MLP architecture, referring to the k-th unit in the l-th layer, we define the corresponding *unit mapping* as

$$f_{l,k}(w_{l,k}, \phi_{l-1}) := \sigma\big(w_{l,k}^\top \phi_{l-1} - b_{l,k}\big), \tag{2}$$

where $\phi_{l-1} \in \mathbb{R}^{n_{l-1}}$ denotes the output from the $(l-1)$-th layer, $w_{l,k} \in \mathbb{R}^{n_{l-1}}$ and $b_{l,k} \in \mathbb{R}$ are respectively a weight vector and a bias associated with the (l, k)-th unit. Note, that the bias $b_{l,k}$ is a free variable in general. However, through our analysis in this work, we fix it as a constant scalar as in Eq. (2) for the sake of convenience for presentation. Then, we can simply define the *l-th layer mapping* by stacking all *unit mappings* in the layer as

$$F_l(W_l, \phi_{l-1}) := \big[f_{l,1}(w_{l,1}, \phi_{l-1}), \ldots, f_{l,n_l}(w_{l,n_l}, \phi_{l-1})\big]^\top, \tag{3}$$

with $W_l := [w_{l,1}, \ldots, w_{l,n_l}] \in \mathbb{R}^{n_{l-1}\times n_l}$ being the l-th weight matrix. Specifically, let us denote by $\phi_0 \in \mathbb{R}^{n_0}$ the input, then the output at the l-th layer is recursively defined as $\phi_l := F_l(W_l, \phi_{l-1})$. Note, that the last layer of

an MLP commonly employs the identity map as the activation function, i.e., $\phi_L := W_L^\top \phi_{L-1}$. Finally, by denoting the set of all parameter matrices in the MLP by $\boldsymbol{\mathcal{W}} := \mathbb{R}^{n_0 \times n_1} \times \ldots \times \mathbb{R}^{n_{L-1} \times n_L}$, we compose all layer-wise mappings to define the overall *MLP network mapping*

$$\begin{aligned} F\colon \boldsymbol{\mathcal{W}} \times \mathbb{R}^{n_0} &\to \mathbb{R}^{n_L}, \\ (\mathbf{W}, \phi_0) &\mapsto F_L(W_L, \cdot) \circ \ldots \circ F_1(W_1, \phi_0). \end{aligned} \tag{4}$$

With such a construction, we can define the set of parameterised mappings specified by a given MLP architecture as

$$\boldsymbol{\mathcal{F}} := \big\{ F(\mathbf{W}, \cdot)\colon \mathbb{R}^{n_0} \to \mathbb{R}^{n_L} \big| \mathbf{W} \in \boldsymbol{\mathcal{W}} \big\}. \tag{5}$$

More specifically, we denote by $\boldsymbol{\mathcal{F}}(n_0, \ldots, n_L)$ the MLP architecture specifying the topology of the MLP, i.e., the number of units in each layer.

Now, let $\boldsymbol{\mathcal{X}} \subseteq \mathbb{R}^{n_0}$ and $\boldsymbol{\mathcal{Y}} \subseteq \mathbb{R}^{n_L}$, one can utilise an MLP $F(\mathbf{W}, \cdot) \in \boldsymbol{\mathcal{F}}$ to approximate the ground truth mapping g^*. Then, an empirical total loss function of MLP based learning can be formulated as

$$\mathcal{J}(\mathbf{W}) := \frac{1}{T} \sum_{i=1}^{T} E\big(F(\mathbf{W}, x_i), y_i\big). \tag{6}$$

If the error function $E(\cdot, \cdot)$ is differentiable in $F(\mathbf{W}, x_i)$, then the function $\mathcal{J}$ is differentiable in the weights $\mathbf{W}$. For the convenience of presentation, in the rest of the paper we denote the sample-wise loss function by

$$J(\mathbf{W}, x_i) := E\big(F(\mathbf{W}, x_i), y_i\big). \tag{7}$$

It is important to notice that, even if the error function $E(\cdot, \cdot)$ is constructed to be convex with respect to the first argument, the *total loss function* $\mathcal{J}$ as defined in Eq. (6) is still *non-convex* in $\mathbf{W}$. Since the set of parameters $\boldsymbol{\mathcal{W}}$ is *unbounded*, when squashing activation functions are employed, exploding the norm of any weight matrix will not drive the function value of $\mathcal{J}$ to infinity. Namely, the total loss function $\mathcal{J}$ can be *non-coercive* [12]. Therefore, the existence and attainability of global minima of $\mathcal{J}$ are not guaranteed in general. However, in the finite sample setting, when appropriate nonlinear activation functions, such as *Sigmoid* and $\tanh$, are employed in the hidden layer, a three-layer MLP with a sufficiently large number of units in the hidden layer can achieve exact learning of finite samples [16, 32].

In the rest of the paper, we assume the existence of an MLP structure that is capable of realising a finite exact approximator.

Assumption 1. *Let $g^*\colon \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}$ be the ground truth mapping. Given T unique samples $\{x_i\}_{i=1}^T \subset \boldsymbol{\mathcal{X}}$, there exists an MLP architecture $\boldsymbol{\mathcal{F}}$, as defined in Eq. (5), and a weight $\mathbf{W}^* \in \boldsymbol{\mathcal{W}}$, so that the corresponding MLP $F(\mathbf{W}^*, \cdot)$ is a finite exact approximator of g^*.*

As exact learning of finite samples is assumed, a suitable error function E is critical to ensure its attainability and uniqueness via an optimisation procedure. Specifically, a finite exact approximator is demanded to correspond with a global minimum of the total loss function without suboptimal local minima. We propose the following assumption as a practical principle of choosing error function.

Principle 1 (Choice of error function)**.** *The error function $E(\cdot, \cdot)$ is differentiable with respect to its first argument. If the gradient of E with respect to the first argument vanishes at $\phi_L \in \mathbb{R}^{n_L}$, i.e., $\nabla_E(\phi_L) = 0$, then ϕ_L is a global minimum of E.*

Remark 1. *Typical examples of error function include the classic squared loss, smooth approximations of ℓ_p norm with $0 < p < 2$, Blake-Zisserman loss, and Cauchy loss [29]. Moreover, by Principle 1, the weights $\mathbf{W}^*$ as assumed in Assumption 1 is a global minimiser of the total loss function $\mathcal{J}$.*

3. Critical point analysis of MLP training

In order to develop a gradient descent algorithm to minimise the cost function $\mathcal{J}$ as in Eq. (6), the derivative of all *unit mappings* are building blocks for our computation. We define the derivative of the activation function σ in the (l, k)-th unit as

$$f'_{l,k}(w_{l,k}, \phi_{l-1}) := \sigma'\left(w_{l,k}^\top \phi_{l-1} - b_{l,k}\right), \tag{8}$$

and the collection of all derivatives of activation functions in the l-th layer as

$$F'_l(W_l, \phi_{l-1}) := \big[f'_{l,1}(w_{l,1}, \phi_{l-1}), \ldots, f'_{l,n_l}(w_{l,n_l}, \phi_{l-1})\big]^\top. \tag{9}$$

For simplicity, we denote by $\phi'_l := F'_l(W_l, \phi_{l-1}) \in \mathbb{R}^{n_l}$. We apply the chain rule of multivariable derivative to compute the directional derivative of J with respect to W_l in direction $H_l \in \mathbb{R}^{n_{l-1} \times n_l}$ as

$$\begin{aligned} \mathrm{D}J(W_l) \cdot H_l = \mathrm{D}E(\phi_L) \cdot \mathrm{D}_2 F_L(W_L, \phi_{L-1}) \cdot \ldots \cdot \\ \cdot \mathrm{D}_2 F_{l+1}(W_{l+1}, \phi_l) \cdot \mathrm{D}_1 F_l(W_l, \phi_{l-1}) \cdot H_l, \end{aligned} \tag{11}$$

where $\mathrm{D}_1 F_l(W_l, \phi_{l-1}) \cdot H_l$ and $\mathrm{D}_2 F_l(W_l, \phi_{l-1}) \cdot h_{l-1}$ refer to the directional derivative of F_l with respect to the first and the second argument, respectively. Explicitly, the first derivative of F_l evaluated at W_l is a linear map $\mathrm{D}_1 F_l(W_l, \phi_{l-1})\colon \mathbb{R}^{n_{l-1} \times n_l} \to \mathbb{R}^{n_l}$, computed as

$$\mathrm{D}_1 F_l(W_l, \phi_{l-1}) \cdot H_l = \mathrm{diag}(\phi'_l) H_l^\top \phi_{l-1}, \tag{12}$$

where the operator $\mathrm{diag}(\cdot)$ puts a vector into a diagonal matrix form, and the first derivative of F_l evaluated at ϕ_{l-1} in direction $h_{l-1} \in \mathbb{R}^{n_{l-1}}$ as

$$\mathrm{D}_2 F_l(W_l, \phi_{l-1}) \cdot h_{l-1} = \mathrm{diag}(\phi'_l) W_l^\top h_{l-1}. \tag{13}$$

The gradient of J in the l-th weight matrix $W_l \in \mathbb{R}^{n_{l-1}\times n_l}$ with respect to the Euclidean metric can be computed as

$$\nabla_J(W_l) = \phi_{l-1}\big(\underbrace{\Sigma_l' W_{l+1}\Sigma_{l+1}' \dots W_L \Sigma_L' \nabla_E(\phi_L)}_{=:\omega_l \in \mathbb{R}^{n_l}}\big)^\top, \quad (14)$$

where $\Sigma_l' := \mathrm{diag}(\phi_l')$. By exploring the layer-wise structure of the MLP, the corresponding vector ω_l can be computed recursively backwards from the output layer L, i.e.,

$$\omega_l := \Sigma_l' W_{l+1}\omega_{l+1}, \quad \text{for all } l = L-1, \dots, 1, \quad (15)$$

with $\omega_L = \Sigma_L' \nabla_E(\phi_L)$. With such a backward mechanism in computing the gradient $\nabla_J(W_l)$, we recover the classic BP algorithm.

In what follows, we characterise critical points of the total loss function $\mathcal{J}$ as defined in Eq. (6) by setting its gradient to zero, i.e., $\nabla_{\mathcal{J}}(\mathbf{W}) = 0$. Explicitly, the gradient of $\mathcal{J}$ with respect to the l-th weight W_l is computed by

$$\nabla_{\mathcal{J}}(W_l) = \sum_{i=1}^{T} \phi_{l-1}\omega_l^\top \in \mathbb{R}^{n_{l-1}\times n_l}, \quad (16)$$

where ϕ_{l-1} and ω_l are respectively the $(l-1)$-th layer output and the l-th error vector as defined in Eq. (15). Similar to the recursive construction of the error vector ω_l as in Eq. (15), we construct a sequence of matrices as, for all $l = L-1, \dots, 1$,

$$\Psi_l := \Sigma_l' W_{l+1}\Psi_{l+1} \in \mathbb{R}^{n_l \times n_L}, \quad (17)$$

with $\Psi_L = \Sigma_L' \in \mathbb{R}^{n_L \times n_L}$. Then the vector form of the gradient $\nabla_{\mathcal{J}}(W_l)$ can be written as

$$\mathrm{vec}\left(\nabla_{\mathcal{J}}(W_l)\right) = \sum_{i=1}^{T} \left(\Psi_l \otimes \phi_{l-l}\right)\nabla_E(\phi_L), \quad (18)$$

where $\mathrm{vec}(\cdot)$ transforms a matrix into a vector by stacking columns on top of one another, and $\otimes$ denotes the Kronecker product of matrices. Then, by applying the previous calculation to all T samples, critical points of the total loss function $\mathcal{J}$ are characterised as solutions of the following parameterised linear system for $\mathbf{W}$

$$\underbrace{\begin{bmatrix} \Psi_L^{(1)}\otimes\phi_{L-1}^{(1)} & \dots & \Psi_L^{(T)}\otimes\phi_{L-l}^{(T)} \\ \vdots & \ddots & \vdots \\ \Psi_1^{(1)}\otimes\phi_0^{(1)} & \dots & \Psi_1^{(T)}\otimes\phi_0^{(T)} \end{bmatrix}}_{=:\mathbf{P}(\mathbf{W})\in\mathbb{R}^{N_{net}\times(Tn_L)}} \underbrace{\begin{bmatrix} \nabla_E(\phi_L^{(1)}) \\ \vdots \\ \nabla_E(\phi_L^{(T)}) \end{bmatrix}}_{=:\boldsymbol{\varepsilon}(\mathbf{W})\in\mathbb{R}^{Tn_L}} = 0, \quad (19)$$

where the superscript $(\cdot)^{(i)}$ indicates the corresponding term for the i-th sample, and $\mathbf{P}(\mathbf{W})$ is the collection of the Jacobian matrices of the MLP for all T samples. Here, N_{net} is the number of variables in the MLP, i.e.,

$$N_{net} = \sum_{l=1}^{L} n_{l-1}n_l. \quad (20)$$

The above parameterised linear equation system in $\boldsymbol{\varepsilon}(\mathbf{W})$ is strongly dependent on several factors, essentially all factors in designing an MLP, i.e., *the MLP structure*, *the activation function*, *the error function*, *given samples*, and *the weight matrices*. If the trivial solution $\boldsymbol{\varepsilon}(\mathbf{W}) = 0$ is reachable at some weights $\mathbf{W}^* \in \boldsymbol{\mathcal{W}}$, then a finite exact approximator $\widehat{g}$ is realised by the corresponding MLP, i.e., $F(\mathbf{W}^*, \cdot) = \widehat{g}$. Additionally, if the solution $\boldsymbol{\varepsilon} = 0$ is the only solution of the parameterised linear equation system for all $\mathbf{W} \in \boldsymbol{\mathcal{W}}$, then any local minimum of the loss function $\mathcal{J}$ is a global minimum. Thus, we conclude the following theorem.

Theorem 1 (suboptimal local minima free condition)**.** *Let an MLP architecture $\boldsymbol{\mathcal{F}}$ satisfy Assumption 1 for a specific learning task, and the error function E satisfy Principle 1. If the following two conditions are fulfilled for all $\mathbf{W} \in \boldsymbol{\mathcal{W}}$,*

(1) the matrix $\mathbf{P}(\mathbf{W})$, as constructed in (19), is non-zero*,*

(2) the vector $\boldsymbol{\varepsilon}(\mathbf{W})$, as constructed in (19), lies in the row span of $\mathbf{P}(\mathbf{W})$,

then a finite exact approximator $\widehat{g}$ is realised at a global minimum $\mathbf{W}^* \in \boldsymbol{\mathcal{W}}$, i.e., $F(\mathbf{W}^*, \cdot) = \widehat{g}$, and the loss function $\mathcal{J}$ is free of suboptimal local minima, i.e., any local minimum of $\mathcal{J}$ is a global minimum.

Obviously, condition (1) in Theorem 1 is quite easy to be ensured, while condition (2) is hardly possible to be realised, since it might require enormous efforts to design the space of MLPs $\boldsymbol{\mathcal{F}}$. Nevertheless, if the rank of matrix $\mathbf{P}$ is equal to Tn_L for all $\mathbf{W} \in \boldsymbol{\mathcal{W}}$, then the trivial solution zero is the only solution of the parameterised linear system. Hence, we have the following proposition as a special case of Theorem 1.

Proposition 1 (Strong suboptimal local minima free condition)**.** *Let an MLP architecture $\boldsymbol{\mathcal{F}}$ satisfy Assumption 1 for a specific learning task, and the error function E satisfy Principle 1. If the rank of matrix $\mathbf{P}(\mathbf{W})$ as constructed in (19) is equal to Tn_L for all $\mathbf{W} \in \boldsymbol{\mathcal{W}}$, then a finite exact approximator $\widehat{g}$ is realised at a global minimum $\mathbf{W}^* \in \boldsymbol{\mathcal{W}}$, i.e., $F(\mathbf{W}^*, \cdot) = \widehat{g}$, and the loss function $\mathcal{J}$ is free of suboptimal local minima.*

Given the number of rows of $\mathbf{P}(\mathbf{W})$ being N_{net}, we suggest the second principle of ensuring performance of MLPs.

Principle 2 (Choice of the number of NN variables)**.** *The total number of variables in an MLP architecture N_{net} needs to be greater than or equal to Tn_L, i.e., $N_{net} \geq Tn_L$.*

In what follows, we investigate the possibility or difficulty to fulfil the condition, i.e., $\mathrm{rank}\left(\mathbf{P}(\mathbf{W})\right) = Tn_L$ for all $\mathbf{W} \in \boldsymbol{\mathcal{W}}$, required in Proposition 1. Let us firstly construct the two identically partitioned matrices ($L \times T$ partitions), by collecting the partitions $\Psi_l^{(i)}$'s and $\phi_l^{(i)}$'s, as

$$\boldsymbol{\Psi} := \begin{bmatrix} \Psi_L^{(1)} \dots \Psi_L^{(T)} \\ \vdots \ \ddots \ \vdots \\ \Psi_1^{(1)} \dots \Psi_1^{(T)} \end{bmatrix}, \text{ and } \boldsymbol{\Phi} := \begin{bmatrix} \phi_{L-1}^{(1)} \dots \phi_{L-1}^{(T)} \\ \vdots \ \ddots \ \vdots \\ \phi_0^{(1)} \ \dots \ \phi_0^{(T)} \end{bmatrix}. \quad (21)$$

Then, the matrix $\mathbf{P}(\mathbf{W})$ constructed on the left hand side of Eq. (19) is computed as the *Khatri-Rao product* of $\mathbf{\Psi}$ and $\mathbf{\Phi}$, i.e., pairwise Kronecker products for all pairs of partitions in $\mathbf{\Psi}$ and $\mathbf{\Phi}$, denoted by

$$\mathbf{P}(\mathbf{W}) := \mathbf{\Psi} \odot \mathbf{\Phi}. \tag{22}$$

Each row block of $\mathbf{P}(\mathbf{W})$ associated with a specific layer l is by construction the *Khatri-Rao product* of the corresponding row blocks in $\mathbf{\Psi}$ and $\mathbf{\Phi}$, i.e., $\mathbf{\Psi}_l \odot \mathbf{\Phi}_{l-1}$ with $\mathbf{\Psi}_l := \left[\Psi_l^{(1)}, \ldots, \Psi_l^{(T)}\right]$ and $\mathbf{\Phi}_{l-1} := \left[\phi_{l-1}^{(1)}, \ldots, \phi_{l-1}^{(T)}\right]$. We firstly investigate the rank property of row blocks of $\mathbf{P}(\mathbf{W})$ in the following proposition[1].

Proposition 2. *Given a collection of matrices $\Psi_i \in \mathbb{R}^{n_l \times n_L}$ and a collection of vectors $\phi_i \in \mathbb{R}^{n_{l-1}}$, for $i = 1, \ldots, T$, let $\mathbf{\Psi} := [\Psi_1, \ldots \Psi_T] \in \mathbb{R}^{n_l \times (n_L T)}$ and $\mathbf{\Phi} = [\phi_1, \ldots, \phi_T] \in \mathbb{R}^{n_{l-1} \times T}$. Then the rank of the Khatri-Rao product $\mathbf{\Psi} \odot \mathbf{\Phi}$ is bounded from below by*

$$\operatorname{rank}(\mathbf{\Psi} \odot \mathbf{\Phi}) \geq n_l \operatorname{rank}(\mathbf{\Phi}) + \sum_{i=1}^{T} \operatorname{rank}(\Psi_i) - T n_l. \tag{23}$$

If all matrices Ψ_i's and $\mathbf{\Phi}$ are of full rank, then the rank of $\mathbf{\Psi} \odot \mathbf{\Phi}$ has the following properties:

(1) If $n_l \leq n_L$, then $\operatorname{rank}(\mathbf{\Psi} \odot \mathbf{\Phi}) \geq n_l \operatorname{rank}(\mathbf{\Phi})$;

(2) If $n_l > n_L$ and $n_{l-1} \geq T$, then $\operatorname{rank}(\mathbf{\Psi} \odot \mathbf{\Phi}) \geq T n_L$;

(3) If $n_l > n_L$ and $n_{l-1} < T$, then $\operatorname{rank}(\mathbf{\Psi} \odot \mathbf{\Phi}) \geq n_L$.

Unfortunately, stacking these row blocks $\mathbf{\Psi}_l \odot \mathbf{\Phi}_{l-1}$ for $l = 1, \ldots, L$ together to construct $\mathbf{P}(\mathbf{W})$ cannot bring better knowledge about the rank of $\mathbf{P}(\mathbf{W})$.

Proposition 3. *For an MLP architecture $\mathcal{F}$, the rank of $\mathbf{P}(\mathbf{W})$ as defined in Eq. (22) is bounded from below by*

$$\begin{aligned} \operatorname{rank}\big(\mathbf{P}(\mathbf{W})\big) \geq & \sum_{l=1}^{L} n_l \operatorname{rank}\left(\mathbf{\Phi}_{l-1}\right) - \sum_{l=1}^{L-1} T n_l \\ & + \sum_{l=1}^{L} \sum_{i=1}^{T} \operatorname{rank}\left(\Psi_l^{(i)}\right) - L T n_L. \end{aligned} \tag{24}$$

Clearly, in order to have the rank of $\mathbf{P}(\mathbf{W})$ sharper bounded from below, it is important to ensure higher rank of all $\Psi_l^{(i)}$'s and $\mathbf{\Phi}_{l-1}$'s. By choosing appropriate activation functions in hidden layers, all $\mathbf{\Phi}_{l-1}$'s can have full rank [16, 32]. Then, in what follows, we present three heuristics aiming to keep all $\Psi_l^{(i)}$'s being of full rank as much as possible. By the construction of $\Psi_l^{(i)}$ as specified in Eq. (17), i.e., matrix product of $\Sigma_l^{'(i)}$'s and $W_l^{(i)}$'s, it is reasonable to ensure full rankness of all $\Sigma_l^{'(i)}$'s and $W_l^{(i)}$'s.

Principle 3 (Constraints on MLP weights)**.** *All weight matrices $W_l \in \mathbb{R}^{n_{l-1} \times n_l}$ for all $l = 1, \ldots, L$ are of full rank.*

[1]The proof is given in the provided supplements.

Remark 2. *Note that, the full rank constraint on the weight matrices does not introduce new local minima of the constrained total loss function. However, whether such a constraint may exclude all global minima of the unconstrained total loss function for a given MLP architecture is still an open problem.*

Principle 4 (Choice of activation functions)**.** *The derivative of activation function σ is non-zero for all $z \in \mathbb{R}$, i.e., $\sigma'(z) \neq 0$.*

Remark 3. *It is trivial to verify that most of popular differentiable activation functions, such as the* Sigmoid, tanh, *SoftPlus, and SoftSign, satisfy Principle 4. Note, that the Identity activation function, which is employed in the output layer, also satisfies this principle. However, potentially vanishing gradient of squashing activation functions can be still an issue in practice due to finite machine precision. Therefore, activation functions without vanishing gradients, e.g. the Bent identity or the leaky ReLU (non-differentiable at the origin), might be preferred.*

However, even if both Principles 3 and 4 are fulfilled, matrices $\Psi_l^{(i)}$'s still cannot be guaranteed to have full rank, according to the Sylvester's rank inequality [33]. Hence, we need to prevent loss of rank in each $\Psi_l^{(i)}$ due to matrix product, i.e., to preserve the smaller rank of two matrices after a matrix product.

Principle 5 (Choice of the number of hidden units)**.** *Given an MLP with hidden layers, the numbers of units in three adjacent layers, namely n_{l-1}, n_l, and n_{l+1}, satisfy the following inequality with $l \leq L-2$*

$$n_l \leq \max\{n_{l-1}, n_{l+1}\}. \tag{25}$$

Remark 4. *The condition $l \leq L-2$ together with $l-1 \geq 0$ implies $L \geq 3$, i.e., the inequality in Eq. (25) takes effect, when there is more than one hidden layer, since Principle 3 and 4 are sufficient to ensure $\Psi_L^{(i)}$'s and $\Psi_{L-1}^{(i)}$'s to have full rank. However, for $l \leq L-2$, the inequality in Eq. (25) together with Principle 3 and 4 ensures no loss of rank in all $\Psi_l^{(i)}$, i.e., $\operatorname{rank}\left(\Psi_l^{(i)}\right) = \min\{n_l, \ldots, n_L\}$.*

Remark 5. *Note, that the pyramidal MLP structure required in [28] follows indeed Principle 5. It can be further verified that the main result of [28] (Theorem 3.8) is a direct application of property (2) in Proposition 2.*

4. Hessian analysis of MLP training

It is well known that gradient descent algorithms can suffer from slow convergence. Information from the Hessian matrix of the loss function is critical for developing efficient numerical algorithms. Specifically, definiteness of the Hessian matrix is an indicator to the isolatedness of the critical points, which will affect significantly the convergence speed of the corresponding algorithm.

We start with the Hessian of the *sample-wise MLP loss function* J as defined in Eq. (7), which is a bilinear operator $\mathsf{H}_J\colon \mathbb{R}^{N_{net}} \times \mathbb{R}^{N_{net}} \to \mathbb{R}$, computed by the second directional derivative of J. For the sake of readability, we only present one component of the second directional derivative with respect to two specific layer indices k and l, i.e.,

$$\begin{aligned} &\mathrm{D}^2 J(\mathbf{W})(H_k, H_l) = \tfrac{\mathrm{d}^2}{\mathrm{d}t^2} J_{l,k}(\mathbf{W}+t\mathbf{H})\big|_{t=0} \\ &= \mathrm{D}^2\, E(\phi_L)\big(\mathrm{D}F(W_l)\cdot H_l, \mathrm{D}F(W_k)\cdot H_k\big) + \\ &\quad + \left(\nabla_E(\phi_L)\right)^\top \mathrm{D}\big(\Sigma'_L W_L^\top \ldots \Sigma'_l H_l^\top \phi_{l-1}\big)\cdot H_k. \end{aligned} \tag{26}$$

Since exact learning of finite samples is assumed at a global minimum $\mathbf{W}^*$, gradients of the error function at all samples are simply zero, i.e., $\nabla_E(\phi_L^{*(i)}) = 0$ for all $i = 1, \ldots, T$. Consequently, the second summand in the last equation in Eq. (26) vanishes for all $i = 1, \ldots, T$. Then, the Hessian $\mathsf{H}_J(\mathbf{W})$ evaluated at $\mathbf{W}^*$ in direction $\mathbf{H} \in \boldsymbol{\mathcal{W}}$ is given as

$$\begin{aligned} &\mathrm{D}^2 J(\mathbf{W}^*)(\mathbf{H},\mathbf{H}) = \mathsf{H}_J(\mathbf{W}^*)(\mathbf{H},\mathbf{H}) \\ &= \sum_{l,k=1}^{L} \big(\mathrm{D}F(W_l^*)\cdot H_l\big)^\top \mathsf{H}_E(\phi_L^*)\big(\mathrm{D}F(W_k^*)\cdot H_k\big), \end{aligned} \tag{27}$$

where $\mathsf{H}_E(\phi_L)\colon \mathbb{R}^{n_L}\times\mathbb{R}^{n_L} \to \mathbb{R}$ is the Hessian of the error function E with respect to the output of the MLP ϕ_L. By a direct computation, we have the Hessian of the total loss function $\mathcal{J}$ at a global minimum $\mathbf{W}^*$ in a matrix form as

$$\mathsf{H}_{\mathcal{J}}(\mathbf{W}^*) = \mathbf{P}(\mathbf{W}^*)\mathbf{H}_E(\mathbf{W}^*)\big(\mathbf{P}(\mathbf{W}^*)\big)^\top, \tag{28}$$

where $\mathbf{P}(\mathbf{W}^*)$ is the Jacobian matrix of the MLP evaluated at $\mathbf{W}^*$ as defined in Eq. (19), and $\mathbf{H}_E(\mathbf{W}^*) := \operatorname{diag}\big(\mathsf{H}_E(\phi_L^{*(1)}), \ldots, \mathsf{H}_E(\phi_L^{*(T)})\big) \in \mathbb{R}^{Tn_L \times Tn_L}$ is a block diagonal matrix of all Hessians of E for all T samples evaluated at $\mathbf{W}^*$. It is then trivial to conclude the following proposition about the rank of $\mathsf{H}_{\mathcal{J}}(\mathbf{W}^*)$.

Proposition 4. *The rank of the Hessian of the total loss function $\mathcal{J}$ at a global minimum $\mathbf{W}^*$ is bounded from above by*

$$\operatorname{rank}(\mathsf{H}_{\mathcal{J}}(\mathbf{W}^*)) \leq Tn_L. \tag{29}$$

Remark 6. *If Principle 2 is assumed, it is easy to see*

$$\operatorname{rank}(\mathsf{H}_J(\mathbf{W}^*)) \leq Tn_L \leq N_{net}. \tag{30}$$

Namely, when an MLP is designed from scratch without insightful knowledge and reaches exact learning, then it is very likely that the Hessian is degenerate, i.e., the classic BP algorithm will suffer significantly from slow convergence. Moreover, Proposition 4 indicates that conditions stated in Corollary 3.9 of [28], i.e., the existence of non-degenerate global minima, is impossible to be satisfied for the setting specified in [28].

If Proposition 1 holds true, i.e., $\operatorname{rank}\big(\mathbf{P}(\mathbf{W}^*)\big) = Tn_L$, then the rank of $\mathsf{H}_{\mathcal{J}}(\mathbf{W}^*)$ will depend on the rank of $\mathbf{H}_E(\mathbf{W}^*)$. To ensure $\mathbf{H}_E(\mathbf{W}^*)$ to have full rank, we need to have a non-degenerate Hessian for the error function E at global minima, i.e., E is a Morse function [26]. Hence, in addition to Principle 1, we state the following principle on the choice of error functions.

Principle 1.a (Strong choice of error function)**.** *In addition to Principle 1, the error function E is Morse, i.e., the Hessian $\mathsf{H}_E(\phi_L)$ is non-degenerate for all $\phi_L \in \mathbb{R}^{n_L}$.*

5. Case studies

In this section, we evaluate our results in the previous sections by two case studies, namely loss surface analysis of training MLPs with one hidden layer, and development of an approximate Newton's algorithm.

5.1. MLPs with one hidden layer

Firstly, we revisit some classic results on MLPs with only one hidden layer [44, 45], and exemplify our analysis using the classic XOR problem [22, 14, 35, 36]. For a learning task with T unique training samples, a finite exact approximator is realisable with a two-layer MLP ($L = 2$) having T units in the hidden layer, and its training process exempts from suboptimal local minima.

Proposition 5. *Let an MLP architecture with one hidden layer satisfy Principle 1, 3, and 4. Then, for a learning task with T unique training samples, if the following two conditions are fulfilled:*

(1) There are T units in the hidden layer, i.e., $n_1 = T$,

(2) T unique samples produce a basis in the output space of the hidden layer for all $W_1 \in \mathbb{R}^{n_0 \times n_1}$,

then a finite exact approximator $\widehat{g}$ is realised at a global minimum $\mathbf{W}^* \in \boldsymbol{\mathcal{W}}$, i.e., $F(\mathbf{W}^*, \cdot) = \widehat{g}$, and the loss function $\mathcal{J}$ is free of suboptimal local minima.

When the scalar-valued bias $b_{l,k}$ in each unit map, as in Eq. (2), is set to be a free variable, a dummy unit is introduced in each layer, except the output layer. The dummy unit always feeds a constant input of one to its successor layer. Results in [44, 45] claim that, with the presence of dummy units, only $T-1$ units in the hidden layer are required to achieve exact learning and eliminate all suboptimal local minima. Such a statement has been shown to be false by counterexample utilising the XOR problem [36]. In the following proposition, we reinvestigate this problem as a concrete example of applying Proposition 3.

Proposition 6. *Let a two-layer MLP architecture $\boldsymbol{\mathcal{F}}(n_0, n_1, n_2)$ with dummy units and $n_2 \leq n_1 \leq T$ satisfy Principle 1, 3, and 4, and $\mathbf{1} := [1, \ldots, 1]^\top \in \mathbb{R}^T$. For a learning task with T unique samples $X \in \mathbb{R}^{n_0 \times T}$, we have*

(1) if $\operatorname{rank}([X^\top, \mathbf{1}]) = n_0$, then

$$\operatorname{rank}\big(\mathbf{P}(\mathbf{W})\big) \geq \max\big\{n_1 n_2, n_1(n_0 + n_2 - T)\big\}; \tag{31}$$

(2) if $\operatorname{rank}([X^\top, \mathbf{1}]) = n_0 + 1$, then

$$\operatorname{rank}\big(\mathbf{P}(\mathbf{W})\big) \geq \max\big\{n_1 n_2, n_1(n_0 + n_2 - T + 1)\big\}. \tag{32}$$

In the rest of this section, we examine these statements on the XOR problem [14, 35, 36]. Two specific MLP structures have been extensively studied in the literature, namely, the $\boldsymbol{\mathcal{F}}(2,2,1)$ network and the $\boldsymbol{\mathcal{F}}(2,3,1)$ network. Squashing activation functions are used in the hidden layer. Dummy units are introduced in both the input and hidden layer. The XOR problem has $T = 4$ unique input samples

$$X := \begin{bmatrix} [\, x_1 & x_2 & x_3 & x_4 \,] \\ [\, 1 & 1 & 1 & 1 \,] \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 4}, \quad (33)$$

where the last row is due to the dummy unit in the input layer and $\mathrm{rank}(X) = 3$. Their associated desired outputs are specified as

$$Y := [y_1\ y_2\ y_3\ y_4] = [0, 1, 1, 0] \in \mathbb{R}^{1\times 4}. \quad (34)$$

Then a squared loss function is defined as

$$\mathcal{J}_{xor}(\mathbf{W}) := \frac{1}{4}\sum_{i=1}^{4} \left(F(\mathbf{W}, x_i) - y_i\right)^2, \quad (35)$$

where $\mathbf{W} := \{W_1, W_2\} \in \mathbb{R}^{3\times n_1} \times \mathbb{R}^{(n_1+1)\times 1}$. In order to identify local minima of the loss function $\mathcal{J}_{xor}$, we firstly compute $n_1(n_0 + n_2 - T + 1) = 0$, according to situation (2) in Proposition 6. Hence, we need to investigate the rank of the Jacobian matrix

$$\mathbf{P}_{xor}(\mathbf{W}) := \begin{bmatrix} 1 \otimes \phi_1^{(1)} & \dots & 1 \otimes \phi_1^{(4)} \\ \Psi_1^{(1)} \otimes x_1 & \dots & \Psi_1^{(4)} \otimes x_4 \end{bmatrix}. \quad (36)$$

Firstly, let us have a look at the $\boldsymbol{\mathcal{F}}(2,2,1)$ XOR network. By considering the dummy unit in the hidden layer, the matrix $\Phi_1 := [\phi_1^{(1)}, \dots, \phi_1^{(4)}] \in \mathbb{R}^{3\times 4}$ has the last row as $[1, 1, 1, 1]$. Then, according to Proposition 2, the first row block in $\mathbf{P}_{xor}$ in Eq. (36) has the smallest rank of two, while the second row block has the smallest rank of one. Hence, the rank of $\mathbf{P}_{xor}(\mathbf{W})$ is lower bounded by two, and local minima can exist for training the $\boldsymbol{\mathcal{F}}(2,2,1)$ XOR network [35, 22].

Similarly, for the $\boldsymbol{\mathcal{F}}(2,3,1)$ network, i.e., $n_1 = 3$, although the first row block in $\mathbf{P}_{xor}$ has potentially the largest rank of four, it is still not immune from collapsing to lower rank of three. Meanwhile, the second row block in $\mathbf{P}_{xor}$ has also the smallest rank of one. Therefore, the rank of $\mathbf{P}_{xor}(\mathbf{W})$ is lower bounded by three. As a sequel, there still exist undesired local minima in training the $\boldsymbol{\mathcal{F}}(2,3,1)$ XOR network [36].

5.2. An approximate Netwon's algorithm

It is important to notice that the Hessian $\mathsf{H}_{\mathcal{J}}(\mathbf{W}^*)$ is neither diagonal nor block diagonal, which differs from the existing approximate strategies of the Hessian in [1, 41]. A classic Newton's method for minimising the total loss function $\mathcal{J}$ requires to compute the exact Hessian of $\mathcal{J}$ from its second directional derivative as in Eq. (26). The complexity of computing the second summand on the right hand side of the last equality in Eq. (26) is of order $O(L^3)$ in the number of layers L. Namely, an implementation of exact Newton's method becomes much more expensive, when an MLP gets deeper. Motivated by the fact that this computationally expensive term vanishes at a global minimum, shown in Eq. (28), we propose to approximate the Hessian of $\mathcal{J}$ at an arbitrary weight $\mathbf{W}$ with the following expression

$$\widetilde{\mathsf{H}}_{\mathcal{J}}(\mathbf{W}) = \mathbf{P}(\mathbf{W})\mathbf{H}_E(\mathbf{W})\big(\mathbf{P}(\mathbf{W})\big)^\top, \quad (37)$$

where $\mathbf{H}_E(\mathbf{W}) := \mathrm{diag}\left(\mathsf{H}_E(\phi_L^{(1)}), \dots, \mathsf{H}_E(\phi_L^{(T)})\right)$, and $\mathbf{P}(\mathbf{W})$ is the Jacobian matrix of the MLP as defined in Eq. (19). With this approximation, we can construct an approximate Newton's algorithm to minimise the total loss function $\mathcal{J}$. Specifically, for the k-th iterate $\mathbf{W}^{(k)}$, an approximate Newton's direction is computed by solving the following linear system for $\boldsymbol{\xi}_N^{(k)} \in \mathbb{R}^{N_{net}}$

$$\widetilde{\mathsf{H}}_{\mathcal{J}}(\mathbf{W}^{(k)})\boldsymbol{\xi}_N^{(k)} = \mathrm{vec}\left(\nabla_{\mathcal{J}}(\mathbf{W}^{(k)})\right), \quad (38)$$

where $\nabla_{\mathcal{J}}(\mathbf{W}^{(k)})$ is the gradient of $\mathcal{J}$ at $\mathbf{W}^{(k)}$. When Principle 1.a holds, the approximate Newton's direction $\boldsymbol{\xi}_N^{(k)}$ can be computed as $\boldsymbol{\xi}_N^{(k)} = \mathbf{P}(\mathbf{W}^{(k)})\boldsymbol{\xi}^{(k)}$ with $\boldsymbol{\xi}^{(k)} \in \mathbb{R}^{Tn_L}$ solving the following linear system

$$\widetilde{\mathsf{H}}_{\mathcal{J}}(\mathbf{W}^{(k)})\mathbf{P}(\mathbf{W}^{(k)})\boldsymbol{\xi}^{(k)} = \mathbf{P}(\mathbf{W}^{(k)})\boldsymbol{\varepsilon}(\mathbf{W}^{(k)}). \quad (39)$$

Then the corresponding Newton's update is defined as

$$\mathrm{vec}(\mathbf{W}^{(k+1)}) = \mathrm{vec}(\mathbf{W}^{(k)}) - \alpha\, \mathbf{P}(\mathbf{W}^{(k)})\boldsymbol{\xi}^{(k)}, \quad (40)$$

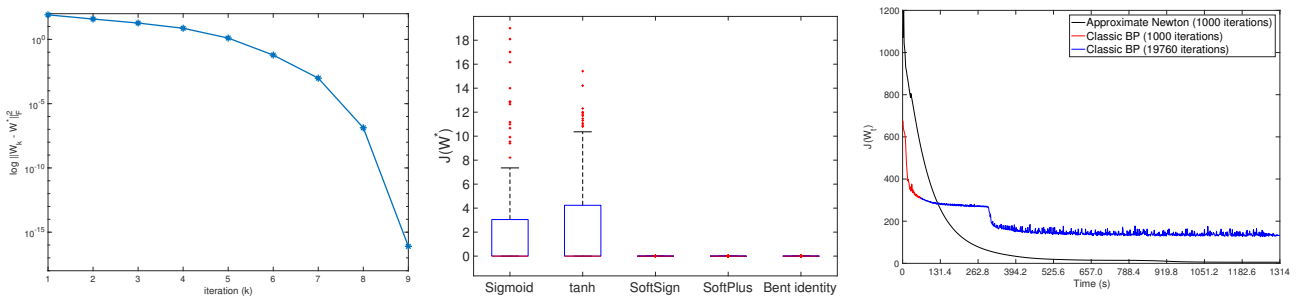
where $\alpha > 0$ is a suitable step size.

Remark 7. *The approximate Hessian proposed in Eq. (37) is by construction positive semi-definite at arbitrary weights, while definiteness of the exact Hessian is not conclusive. It is trivial to see that the approximate Hessian coincides with the ground-truth Hessian as Eq. (28) at global minima. Hence, when $\alpha = 1$, the corresponding approximate Newton's algorithm induced by the update rule in Eq. (40) shares the same local quadratic convergence properties to a global minimum as the exact Newton's method (see Section 6).*

In general, computing the approximate Newton's update as defined in Eq. (38) can be computationally expensive. Interestingly, the approximate Newton's algorithm is indeed the state of the art GGN algorithm developed in [31]. Efficient implementations of the GGN algorithm have been extensively explored in [23, 7, 4]. In the next section, we investigate the theoretical convergence properties of the approximate Newton's algorithm, i.e., the GGN algorithm.

6. Numerical evaluation

In this section, we investigate performance of the Approximate Newton's (AN) algorithm, i.e., the GGN algorithm. We test the algorithm on the four regions classification benchmark, as originally proposed in [34]. In $\mathbb{R}^2$



(a) *Convergence with exact learning* (b) *Impact of vanishing gradient* (c) *Convergence without exact learning*

Figure 1. Investigation of an approximate Newton's algorithm, a.k.a. the Generalised Gauss-Newton algorithm.

around the origin, we have a square area $(-4, 4) \times (-4, 4)$, and three concentric circles with their radiuses being 1, 2, and 3. Four regions/classes are interlocked and nonconvex, see [34] for further details of the benchmark. Samples are drawn in the box for training with their corresponding outputs being the classes $\{1, 2, 3, 4\}$.

We firstly demonstrate theoretical convergence properties of the AN/GGN algorithm with $\alpha = 1$, by deploying an MLP architecture $\mathcal{F}(2, 20, 1)$. Dummy units are introduced in both the input and hidden layer. Activation functions in the hidden layer are chosen to be the *Bend identity*, and the error function is the squared loss. With a set of 20 randomly drawn training samples, we run the AN/GGN algorithm from randomly initialised weights. The convergence is measured by the distance of the accumulation point $\mathbf{W}^*$ to all iterates $\mathbf{W}^{(k)}$, i.e., by an extension of the Frobenius norm of matrices to collections of matrices as $\|\mathbf{W}^{(k)} - \mathbf{W}^*\|_F^2 := \sum_{l=1}^{L} \|W_l^{(k)} - W_l^*\|_F^2$. It is clear from Figure 1(a) that the AN/GGN algorithm converges locally quadratically fast to a global minimiser of exact learning.

We further investigate the performance of AN/GGN with five different activations, namely *Sigmoid*, $\tanh$, *SoftSign*, *SoftPlus*, and *Bent identity*. The first three activation functions are squashing, while the SoftPlus is only bounded from below, and the Bent identity is totally unbounded. Figure 1(b) gives the box plot of the value of the total loss function $\mathcal{J}$ over 100 independent runs from random initialisations after convergence. Clearly, AN/GGNs with SoftPlus and Bent identity perform very well, while the ones with Sigmoid and $\tanh$ suffer from numerically spurious local minima. However, it is also very interesting to observe that SoftSign does not share the bad convergence behaviour as its squashing counterparts, due to some unknown factors.

Finally, we investigate the AN/GGN algorithm, comparing with the classic BP algorithm in terms of convergence speed, without exact learning being assumed. In this experiment, we adopt an MLP architecture $\mathcal{F}(2, 10, 10, 4)$, where the target outputs being the corresponding standard basis vector in $\mathbb{R}^4$, and the set step size to be constant $\alpha = 0.01$. For running 1000 iterations, the BP algorithm took 61.1 sec., while the AN/GGN algorithm spent 1314.1 sec. On average, the running time for each iteration of AN/GGN was about 21.4 times as required for an iteration of BP. With the same data and the same random initialisation, we ran BP for 20760 iterations, which took the same amount of time as required for 1000 iterations of AN/GGN. As shown in Figure 1(c), the first 1000 iterations of BP was highlighted in *red* with the remaining iterations being coloured in *blue*. The AN/GGN goes up at the beginning, then smoothly converges to the global minimal value, while the BP demonstrates strong oscillation towards the end.

7. Conclusion

In this work, we provide a smooth optimisation perspective on the challenge of training MLPs. Under the condition of exact learning, we characterise the critical point conditions of the empirical total loss function, and investigate sufficient conditions to ensure any local minimum to be globally minimal. Classic results on MLPs with only one hidden layer are reexamined in the proposed framework. Finally, the so-called Generalised Gauss-Newton algorithm is rigorously revisited as an approximate Newton's algorithm, which shares the property of being locally quadratically convergent to a global minimum. All aspects discussed in this paper require a further systematic and thorough investigation both theoretically and experimentally, and are expected to be also applicable for training recurrent neural networks.

Furthermore, it is worth noticing that the present work only considers the case of exact learning of MLPs. When training data is contaminated by noise or outlier, exact learning can be more problematic than an approximate learning, due to overfitting. Since exact learning needs to be avoided in this case, and the loss function of MLP training is non-coercive in general, it is crucial as well as difficult to select appropriate regularisers that ensure both generalisability and solvability of regularised MLP training.

References

[1] R. Battiti. First- and second-order methods for learning: Between steepest descent and newton's method. *Neural Computation*, 4(2):141–166, 1992. 2, 7

[2] C. Bishop. Exact calculation of the hessian matrix for the multilayer perceptron. *Neural Computation*, 4(4):494–501, 1992. 2

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1996. 1

[4] A. Botev, H. Ritter, and D. Barber. Practical gauss-newton optimisation for deep learning. In *Proceedings of the* 34^{th} *International Conference on Machine Learning*, 2017. 7

[5] C. Charalambous. Conjugate gradient algorithm for efficient training of artificial neural networks. *IEE Proceedings G - Circuits, Devices and Systems*, 139(3):301–310, 1992. 2

[6] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Proceedings of the* 18^{th} *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 192–204, 2015. 1

[7] M. Fairbank and E. Alonso. Efficient calculation of the Gauss-Newton approximation of the Hessian matrix in neural networks. *Neural Computation*, 24(3):607–610, 2012. 7

[8] T. Falas and A. G. Stafylopatis. The impact of the error function selection in neural network-based classifiers. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 1799–1804, 1999. 1

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the* 13^{th} *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 249–256, 2010. 1

[10] I. J. Goodfellow, O. Vinyals, and A. M. Saxe. Qualitatively characterizing neural network optimization problems. Published at the 5^{th} International Conference on Learning Representations (ICLR). arXiv:1412.6544., 2015. 1

[11] M. Gori and A. Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992. 1

[12] O. Güler. *Foundations of Optimization*. Springer, 2010. 3

[13] B. D. Haeffele and R. Vidal. Global optimality in neural network training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7331 – 7339, 2017. 1, 2

[14] L. G. C. Hamey. XOR has no local minima: A case study in neural network error surface analysis. *Neural Metworks*, 11(4):669–681, 1998. 1, 6, 7

[15] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991. 1

[16] Y. Ito. Nonlinearity creates linear independence. *Advances in Computational Mathematics*, 5(1):189–203, 1996. 3, 5

[17] K. Kawaguchi. Deep learning without poor local minima. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 586–594, 2016. 1

[18] D. P. Kingma and J. L. Ba. Adam: a method for stochastic optimization. In *The* 3*-rd International Conference on Learning Representations*, pages 1–13, 2015. 2

[19] J. F. Kolen and J. B. Pollack. Backpropagation is sensitive to initial conditions. *Complex Systems*, 4(3):269–280, 1990. 1

[20] Q. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Ng. On optimization methods for deep learning. Proceedings of international conference on Machine Learning, 2011. 2

[21] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015. 1

[22] P. J. G. Lisboa and S. J. Perantonis. Complete solution of the local minima in the XOR problem. *Network: Computation in Neural Systems*, 2(1):119–124, 1991. 1, 6, 7

[23] J. Martens. Deep learning via Hessian-free optimization. In *Proceedings of the* 27^{th} *International Conference on Machine Learning (ICML)*, pages 735–742, 2010. 2, 7

[24] J. Martens. New perspectives on the natural gradient method. arXiv:1412.1193v8, 2014. 2

[25] H. N. Mhaskar and C. A. Micchelli. How to choose an activation function. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, pages 319–326, 1993. 1

[26] J. Milnor. *Morse Theory*. Princeton University Press, 1963. 6

[27] E. Mizutani and S. E. Dreyfus. Second-order stagewise backpropagation for hessian-matrix analyses and investigation of negative curvature. *Neural Networks*, 21(2-3):193–203, 2008. 2

[28] Q. Nguyen and M. Hein. The loss surface of deep and wide neural networks. In *Proceedings of the* 34^{th} *International Conference on Machine Learning*, 2017. 1, 5, 6

[29] H. R. and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, 2004. 3

[30] R. Rojas. *Neural Networks: A Systematic Introduction*. Springer, 1996. 1

[31] N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14:1723 – 1738, 2002. 2, 7

[32] J. V. Shah and C.-S. Poon. Linear independence of internal representations in multilayer perceptrons. *IEEE Transactions on Neural Networks*, 10(1):10–18, 1999. 3, 5

[33] D. A. Simovici. *Linear Algebra Tools for Data Mining*. World Scientific Publishing Company, 2012. 5

[34] S. Singhal and L. Wu. Training multilayer perceptrons with the extended Kalman algorithm. In *Advances in Neural Information Processing Systems*, pages 133–140, 1989. 7, 8

[35] I. G. Sprinkhuizen-Kuyper and E. J. W. Boers. The local minima of the error surface of the 2-2-1 XOR network. *Annals of Mathematics and Artificial Intelligence*, 25(1):107–136, 1999. 1, 6, 7

[36] I. G. Sprinkhuizen-Kuyper and E. J. W. Boers. A local minimum for the 2-3-1 XOR network. *IEEE Transactions on Neural Networks*, 10(4):968–971, 1999. 1, 6, 7

[37] S. Sun, W. Chen, L. Wang, X. Liu, and T.-Y. Liu. On the depth of deep neural networks: A theoretical view. In *Proceedings of the* 31^{st} *AAAI Conference on Artificial Intelligence*, pages 2066–2072, 2016. 1

[38] R. S. Sutton. Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proceedings of the* 8*-th Annual Conference of the Cognitive Science Society*, pages 823–831, 1986. 1

[39] O. Vinyals and D. Povey. Krylov subspace descent for deep learning. In *Proceedings of the* 15^{th} *International Conference on Arti cial Intelligence and Statistics (AISTATS)*, volume 22, pages 1261–1268, 2012. 2

[40] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon. Accelerating the convergence of the backpropagation method. *Biological Cybernetics*, 59(4):257–263, 1988. 2

[41] Y.-J. Wang and C.-T. Lin. A second-order learning algorithm for multilayer networks based on block Hessian matrix. *Neural Networks*, 11(9):1607–1622, 1998. 2, 7

[42] B. Widrow and M. A. Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990. 1

[43] D. Yu and L. Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer-Verlag, London, 2015. 1

[44] X.-H. Yu. Can backpropagation error surface not have local minima. *IEEE Transactions on Neural Networks*, 3(6):1019–1021, 1992. 1, 6

[45] X.-H. Yu and G.-A. Chen. On the local minima free condition of backpropagation learning. *IEEE Transactions on Neural Networks*, 6(5):1300–1303, 1995. 1, 6