2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

First order online optimisation using forward gradients in over-parameterised systems

Behnam Mafakheri[1], Jonathan H. Manton[1] and Iman Shames[2]

***Abstract*— The success of deep learning over the past decade mainly relies on gradient-based optimisation and backpropagation. This paper focuses on analysing the performance of first-order gradient-based optimisation algorithms with time-varying non-convex cost function under Polyak-Łojasiewicz condition. Specifically, we focus on using the forward mode of automatic differentiation to compute directional derivatives of the cost function in fast-changing problems where calculating gradients using the backpropagation algorithm is either impossible or inefficient. Upper bounds for tracking and asymptotic errors are derived for various cases, showing the linear convergence to a solution or a neighbourhood of an optimal solution, where the convergence rate decreases with the increase in the dimension of the problem. We present numerical results demonstrating the method's correctness and performance.**

***Index Terms*— Online optimisation, Directional derivatives, Over-parameterised systems, PL condition**

I. Introduction

Gradient-based optimisation is the core of most machine learning algorithms [11]. Backpropagation, the reverse mode of Automatic Differentiation algorithms, has been the main algorithm for computing gradients. In some cases, using backpropagation is either impossible or inefficient (see [22] on the motivations for using directional derivatives and gradient-free method). In [3] it is argued that in training neural networks, one can calculate gradients based on directional derivatives, which is faster than backpropagation. Note that for a real-valued function, a directional derivative is only a scalar while gradient is a vector. Directional derivatives can be calculated using a single forward evaluation of a function without the necessity of storing the results of all intermediate computations, these make calculating directional derivatives more desirable in resource-limited applications. An unbiased estimate of gradient, named forward gradient, can speed up training up to twice as fast in some examples [2].

Given a function $f : \mathbb{R}^n \to \mathbb{R}$, the *forward gradient* at point $\boldsymbol{x}$ is defined as $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = \langle \nabla f(\boldsymbol{x}), \boldsymbol{u} \rangle \boldsymbol{u}$, where $\boldsymbol{u}$ is a random vector with zero mean and unit variance. Note that $\langle \nabla f(\boldsymbol{x}), \boldsymbol{u} \rangle$ is the directional derivative of f at point $\boldsymbol{x}$ in the direction $\boldsymbol{u}$. The forward gradient, $g(\boldsymbol{x}, \boldsymbol{u})$, is an unbiased estimator for $\nabla f(\boldsymbol{x})$ as long as the components of direction $\boldsymbol{u}$ are independent and sampled from a distribution with zero mean and unit variance; $[\mathbb{E}(g(\boldsymbol{x}, \boldsymbol{u}))]_i = \mathbb{E}(u_i \sum_{j=1}^{n} (\nabla f(\boldsymbol{x}))_j u_j) = (\nabla f(\boldsymbol{x}))_i$, where a_i is the i-th element of vector $\boldsymbol{a}$.

Deep neural networks can be considered as a particular case of the over-parameterised system of nonlinear equations in which there are potentially a staggering number of trainable parameters [9]. In [19] authors argue that focusing on the convexity of the problem does not lead to a suitable framework for analysing such over-parameterised loss functions and instead show that the Polyak-Łojasiewicz (PL) condition [25] is satisfied on the most of parameter space. This, in turn, explains the fast convergence of the Gradient Descent (GD) algorithm to a global minimum.

Another area that has observed a flurry of activity is anchored at studying time-varying objective functions. These problems arise in many applications, including but not limited to robotics, smart grids, communication systems and signal processing. An example is when a machine learning problem's data arrive sequentially in a stream. These can be considered as a sequence of optimisation problems. A solver may solve each problem completely before approaching the next problem, which is not feasible in cases where each instance of the problem is of large scale or involves communication over a network. Therefore, a more efficient way may be to solve each problem partially (e.g. performing only a few gradient descent steps) before receiving the next problem. Unlike batch learning, which requires the entire training data set to be made available before the learning task, online learning represents a class of algorithms that learn to optimise predictive models over a stream of data instances sequentially. Ideally, the solution generated by the method at each time step, $\boldsymbol{x}_k$, follows $\boldsymbol{x}_k^*$. In other words, the tracking error $\|\boldsymbol{x}_k - \boldsymbol{x}_k^*\|$ or the instantaneous sub-optimality gap (a.k.a *instantaneous regret*), $\|\mathcal{L}_k(\boldsymbol{x}_k) - \mathcal{L}_k(\boldsymbol{x}_k^*)\|$, either shrink or remain bounded as k grows. Therefore, we use these performance measures for the algorithms in this paper. We refer to Section II for the discussion on how this is related to different kinds of performance measures in the context of online learning.

In this paper, we focus on the analysis of the performance of the forward gradient-based method in time-varying settings characterised by the following cost function at time k:

$$\min_{\boldsymbol{x} \in \mathbb{R}^m} \{\mathcal{L}_k(\boldsymbol{x}) := g_k(\boldsymbol{x}) + h_k(\boldsymbol{x})\}. \tag{1}$$

The function g_k is smooth and potentially nonconvex and $h_k(\boldsymbol{x})$ is possibly a nonsmooth convex function that imposes some structure on the solution. Such problems arise in online machine learning [27]. The contributions of this paper are listed below:

*This work was supported by the Australian Research Council under the Discovery Projects funding scheme (DP210102454).

[1]B. Mafakheri and J. H. Manton are with Department of Electrical and Electronic Engineering, University of Melbourne, VIC 3010, Australia {mafakherib, jmanton}@unimelb.edu.au

[2]I. Shames is with the The CIICADA Lab, School of Engineering, The Australian National University, Canberra, ACT 2601, Australia iman.shames@anu.edu.au

- We prove the linear convergence in expectation of forward-gradient method to the minimiser for smooth nonconvex PL objective functions (Theorem 8).
- We prove that the expected tracking error of the forward-gradient method in a time-varying nonconvex setting, under smoothness and PL assumptions, converges to a neighbourhood of zero at a linear rate (Theorem 16).
- We demonstrate that using directional derivatives are effective even in high dimension for a pre-trained neural network that observes new data samples at a high rate while operating (see Remark 17).

The rest of the paper is organised as follows: in Section II, we discuss related works, and in Section III, we review notations, definitions, basic assumptions and some background knowledge. In Section IV-A, we prove the linear convergence of the forward gradient algorithm for the PL objective functions. In Section IV-B, we analyse the performance of the method based on forward gradients under PL condition. Section V demonstrates numerical examples and Section VI concludes the paper.

II. Related works

PL condition in over-parameterised systems: The nonlinear least square problem has been extensively studied in under-parameterised systems [23]. In [19, 4] a sufficient condition for satisfying the PL condition in a wide neural network with the least square loss function and linear convergence of Stochastic Gradient Descent (SGD) has been studied and the relation with neural tangent kernel has been discussed [13].

Static cost functions: In [15] convergence of Randomised Coordinate Descent, SGD, and SVRG algorithms under PL condition and Proximal coordinate descent algorithm under Proximal-PL has been studied. Inexact gradient descent algorithms have been analysed under various assumptions (See [16] and references therein). In [30] $\mathcal{O}(1/k)$ convergence of SGD under weaker conditions (Weak Growth Condition) has been proved.

Time-varying cost function: A closely related line of research to this paper is online convex optimisation (OCO) which was originally introduced in seminal work by [33]. See [12] for a comprehensive review of OCO methods. Algorithmic performance is evaluated in terms of the *stationary regret*, defined as the integrated difference between the observed cost and the cost at the best fixed decision variable. *Dynamic regret* is a generalisation which compares the observed cost with the instantaneous optimal cost. OCO results are typically presented in terms of regret bounds, the goal being to achieve sub-linear regret (see [21] and references therein). In [32, 8, 26, 31] the regret bounds are derived for online first-order optimisation algorithms under (strong) convexity and smoothness with noisy gradients. Regret analysis of online stochastic fixed and time-varying optimisation have been done in [12, 7, 28]. The works in [24] and [17] have analysed the online (proximal) gradient methods with sub-Weibull gradient error under strong convexity and PL conditions, respectively. In [10], assuming bounded total temporal change of the sequence of loss functions, a sublinear regret bound for a class of nonconvex functions has been established. By assuming access to an offline optimisation oracle, sublinear regret bounds for classical Follow the Perturbed Leader (FTPL) have been shown in [1] and [29]. The previous works all prove bounds on static regret. In contrast, we focus on tracking error bound and instantaneous regret from which the dynamic regret can be analysed as a by-product.

III. Notation and Preliminaries

Throughout this paper, we show the set of real numbers as $\mathbb{R}$. For vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ the Euclidean inner product and its corresponding norms are denoted by $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ and $\|\boldsymbol{a}\|$ respectively. We denote the open ball centred at $\boldsymbol{x}$ with radius r by $B(\boldsymbol{x}, r)$. With $\pi_{\mathcal{X}}(\boldsymbol{x})$ we indicate the projection of vector $\boldsymbol{x}$ into set $\mathcal{X}$. The Euclidean distance to a set $\mathcal{X} \in \mathbb{R}^n$ is defined by $dist(\boldsymbol{x}, \mathcal{X}) := \inf_{\boldsymbol{u} \in \mathcal{X}} \|\boldsymbol{u} - \boldsymbol{x}\|$ for every $\boldsymbol{x} \in \mathbb{R}^n$. We denote the mapping of projection over set $\mathcal{X}$ with $\pi_{\mathcal{X}}$. With $\mathcal{X}^*$ we denote the set of minimisers of a function. For a matrix $A \in \mathbb{R}^{m \times n}$, $\|A\|$ is the spectral norms of the matrix. We use $\mathcal{D}F(\cdot)$ and $\mathcal{D}^2F(\cdot)$ to show the derivative and Hessian of function F, respectively. We use $\widetilde{\mathcal{O}}(\cdot)$ notation by ignoring the logarithmic factors in big-O notation, i.e. $\widetilde{\mathcal{O}}(f(n)) = \mathcal{O}(f(n) \log f(n))$.

In what follows, we introduce some definitions, assumptions and lemmas that will be utilised in the paper.

Definition 1: The function $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ is $\beta - smooth$ if it is differentiable and

$$\|\nabla \mathcal{L}(\boldsymbol{y}) - \nabla \mathcal{L}(\boldsymbol{x})\| \leq \beta \|\boldsymbol{y} - \boldsymbol{x}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n.$$

Definition 2: The function $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ satisfies the Polyak-Łojasiewicz (PL) condition on a set $\mathcal{X}$ with constant μ if

$$\|\nabla \mathcal{L}(\boldsymbol{x})\|^2 \geq 2\mu(\mathcal{L}(\boldsymbol{x}) - \mathcal{L}^*), \quad \forall \boldsymbol{x} \in \mathcal{X}, \tag{2}$$

where $\mathcal{L}^\star$ is the global minimum of $\mathcal{L}(\boldsymbol{x})$.

Lemma 3: (Descent Lemma [5, Proposition A.24]) Let the function $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ be a $\beta_{\mathcal{L}}$-smooth function. Then for every $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and for every $\boldsymbol{z} \in [\boldsymbol{x}, \boldsymbol{y}] := \{(1-\alpha)\boldsymbol{x} + \alpha \boldsymbol{y} : \alpha \in [0, 1]\}$ the following holds

$$\mathcal{L}(\boldsymbol{y}) \leq \mathcal{L}(\boldsymbol{x}) + \langle \nabla \mathcal{L}(\boldsymbol{z}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{\beta_{\mathcal{L}}}{2} \|\boldsymbol{y} - \boldsymbol{x}\|^2.$$

Lemma 4: ([15]) Let $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ be a β-smooth and $\mu - PL$ function. Then

$$\frac{\mu}{2} \|\boldsymbol{x} - \pi_{\mathcal{X}^*}(\boldsymbol{x})\|^2 \leq \mathcal{L}(\boldsymbol{x}) - \mathcal{L}^* \leq \frac{\beta}{2} \|\boldsymbol{x} - \pi_{\mathcal{X}^*}(\boldsymbol{x})\|^2, \quad \forall \boldsymbol{x}.$$

Definition 5: (Proximal-PL condition [15]) Let $\mathcal{L}(\boldsymbol{x}) = g(\boldsymbol{x}) + h(\boldsymbol{x})$ where g is $\beta-$smooth and h is a convex function. The function f satisfies $\mu-$Proximal PL-condition if the following holds

$$\mathcal{D}_h(\boldsymbol{x}, \beta) \geq 2\mu(\mathcal{L}(\boldsymbol{x}) - \mathcal{L}^*), \quad \forall \boldsymbol{x}, \tag{3}$$

where

$$\mathcal{D}_h(\boldsymbol{x}, \alpha) := -2\alpha \min_{\boldsymbol{y}} \{\langle \nabla g(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{\alpha}{2} \|\boldsymbol{y} - \boldsymbol{x}\|^2 + h(\boldsymbol{y}) - h(\boldsymbol{x})\}. \tag{4}$$

A. Over-parameterised systems and PL condition

For a system of n nonlinear equations

$$f(\boldsymbol{x};\boldsymbol{a}_i)=b_i,\quad i=1,2,\dots,n,$$

where $\{\boldsymbol{a}_i,b_i\}_{i=1}^n$ is the set of model parameters and one aims at finding $\boldsymbol{x}\in\mathbb{R}^m$ that solves the system of equations. Aggregating all equations in a single map amounts to

$$\mathcal{F}(\boldsymbol{x})=\boldsymbol{y},\ \text{where } \boldsymbol{x}\in\mathbb{R}^m,\mathcal{F}(\cdot):\mathbb{R}^m\to\mathbb{R}^n. \tag{5}$$

The system in (5) is solved through minimising a certain loss function $\mathcal{L}(\boldsymbol{x}):=\frac{1}{2}\|\mathcal{F}(\boldsymbol{x})-\boldsymbol{b}\|^2=\frac{1}{2}\sum_{i=1}^n(f(\boldsymbol{x};\boldsymbol{a}_i)-b_i)^2$. This problem has been studied extensively in under-parameterised settings (where $m<n$). We refer to the exact solution of (5) as interpolation.

Definition 6: Let $D\mathcal{F}(\boldsymbol{x})$ be the differential of the map $\mathcal{F}$ at $\boldsymbol{x}$, which can be represented as a $n\times m$ matrix. The tangent Kernel of $\mathcal{F}$ is defined as a $n\times n$ positive semidefinite matrix $K(\boldsymbol{x}):=D\mathcal{F}(\boldsymbol{x})D\mathcal{F}^T(\boldsymbol{x})$.

Definition 7: We say that a non-negative function $\mathcal{L}$ satisfies $\mu-PL^*$ condition on a set $\mathcal{X}\in\mathbb{R}^m$ for $\mu>0$, if

$$\|\nabla\mathcal{L}(\boldsymbol{x})\|^2\geq 2\mu\mathcal{L}(\boldsymbol{x}),\quad \forall\boldsymbol{x}\in\mathcal{X}. \tag{6}$$

Remark 8: (a) if a non-negative function satisfies $\mu-PL^*$, then it will satisfy $\mu-PL$ condition too (2).

(b) Every stationary point of a PL function is a global minimum.

The following results are of great importance in the later discussions.

Theorem 9: ([25]) If $\mathcal{L}(\boldsymbol{x})$ is lower bounded with β-Lipschitz continuous gradients and satisfies $\mu-PL$ condition in the region $\mathcal{X}=\overline{B(\boldsymbol{x}_0,\rho)}$ where $\rho>\frac{1}{\mu}\sqrt{2\beta(\mathcal{L}(\boldsymbol{x}_0)-\mathcal{L}^*)}$, then there exists a global minimum point $x^*\in\mathcal{X}$ and the gradient descent algorithm with a small enough step-size ($\frac{1}{\beta}$) converges to x^* in a linear rate.

Theorem 10: ([19]) If $\mathcal{F}(\boldsymbol{x})$ is such that $\lambda_{min}(K(\boldsymbol{x}))\geq\mu>0$ for all $\boldsymbol{x}\in\mathcal{X}$, then the square loss function $\mathcal{L}(\boldsymbol{x})=\frac{1}{2}\|\mathcal{F}(\boldsymbol{x})-\boldsymbol{y}\|^2$ satisfies $\mu-PL^*$ condition on $\mathcal{X}$.

Note that $K(\boldsymbol{x})=D\mathcal{F}(\boldsymbol{x})D\mathcal{F}^T(\boldsymbol{x})$ implies that $\text{rank}(K(\boldsymbol{x}))=\text{rank}(D\mathcal{F}(\boldsymbol{x}))$. In an over-parameterised system, $m>n$, starting from a random initial point, there is a high probability that $D\mathcal{F}(\boldsymbol{x})$ is full rank. However, from Theorem 9, one needs $\mu-PL^*$ condition in a large enough ball with radius $\mathcal{O}(\frac{1}{\mu})$ for linear convergence of the GD algorithm. For establishing such conditions, one intuitively can expect that in the cases of $\mathcal{C}^2$ function, if Hessian (curvature) is small enough in the neighbourhood of a point, then the Tangent Kernel should be almost constant and therefore, the conditions for linear convergence of the GD algorithm will be satisfied. In the case of highly over-parameterised Neural Networks (wide NNs) with linear output layer, the Hessian matrix will have arbitrarily small spectral norm (a transition to linearity). This is formulated as follows:

Theorem 11: ([20]) For an L layer neural network with a linear output layer and minimum width m of the hidden layers, for any $R>0$ and any $\boldsymbol{x}\in B_R(\boldsymbol{x}_0)$. the Hessian spectral norm satisfies $\|\mathcal{D}^2\mathcal{F}(\boldsymbol{x})\|=\widetilde{\mathcal{O}}\left(\frac{R^{3L}}{\sqrt{m}}\right)$ with a high probability.

All in all, wide neural networks, as a particular case of over-parameterised systems, satisfy the $\mu-PL^*$ condition, which explains the fast convergence of (S)GD to a global minimum in square-loss problems. In the following sections, we theoretically analyse the performance of using the forward gradient in the same setting (over-parameterised systems).

IV. Optimisation using forward gradient

In this section, we analyse the performance of various gradient-based algorithms using forward gradient.

A. Forward gradient for fixed cost functions

With the focus on the basic unconstrained optimisation problem $\min_{\boldsymbol{x}\in\mathbb{R}^m}\mathcal{L}(x)$. Consider the following iterations where the solver is not fast enough to do gradient calculation, but it can do a forward gradient update at time k,

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k-\alpha_k\boldsymbol{v}_k(\boldsymbol{x}_k,U_k), \tag{7}$$

where α_k is step-size, $\boldsymbol{v}(\boldsymbol{x},\boldsymbol{u})=\langle\nabla\mathcal{L}(\boldsymbol{x}),\boldsymbol{u}\rangle\boldsymbol{u}$, and $U_k\sim\mathcal{N}(\boldsymbol{0},I_m)$ are i.i.d random directions for all k, which implies that $\boldsymbol{v}(\boldsymbol{x},\boldsymbol{u})$ is an unbiased estimator of $\nabla\mathcal{L}(\boldsymbol{x})$. It is extensively studied in [22] where the following properties have been proved:

$$\mathbb{E}\boldsymbol{v}(\boldsymbol{x},U_k)=\nabla\mathcal{L}(\boldsymbol{x}), \tag{8a}$$

$$\mathbb{E}(\|\boldsymbol{v}(\boldsymbol{x},U_k)\|^2)\leq(m+4)\|\nabla\mathcal{L}(\boldsymbol{x})\|^2. \tag{8b}$$

In the following, we analyse the convergence of this algorithm under PL condition.

Theorem 12: Assume that function $\mathcal{L}$ is $\beta-$smooth, has a non-empty solution set $\mathcal{X}^*$, and satisfies $\mu-$PL condition. Consider the algorithm (7) with a step-size of $\frac{1}{\beta(m+4)}$. If random vector U_k is chosen from a standard normal distribution, i.e. $U_k\sim\mathcal{N}(\boldsymbol{0},I_m)$, then the algorithm has an expected linear convergence rate

$$\mathbb{E}\{\mathcal{L}(x_k)-\mathcal{L}^*\}\leq\left(1-\frac{\mu}{(m+4)\beta}\right)^k(\mathcal{L}(x_0)-\mathcal{L}^*).$$

Proof: Using the descent lemma we have

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}_{k+1})&\leq\mathcal{L}(\boldsymbol{x}_k)+\langle\nabla\mathcal{L}(\boldsymbol{x}_k),\boldsymbol{x}_{k+1}-\boldsymbol{x}_k\rangle\\&\quad+\frac{\beta}{2}\|\boldsymbol{x}_{k+1}-\boldsymbol{x}_k\|^2\\&=\mathcal{L}(\boldsymbol{x}_k)-\alpha_k\langle\nabla\mathcal{L}(\boldsymbol{x}_k),\boldsymbol{v}_k(\boldsymbol{x}_k,U_k)\rangle+\frac{\beta}{2}\alpha_k^2\|\boldsymbol{v}_k(\boldsymbol{x}_k,U_k)\|^2\end{aligned}$$

By taking conditional expectation given $\boldsymbol{x}_k$ with respect to U_k, and using (8) we obtain

$$\begin{aligned}&\mathbb{E}\{\mathcal{L}(\boldsymbol{x}_{k+1})\mid\boldsymbol{x}_k\}-\mathcal{L}(\boldsymbol{x}_k)\leq-\alpha_k\|\nabla\mathcal{L}(\boldsymbol{x}_k)\|^2\\&+\frac{1}{2}\beta\alpha_k^2(m+4)\|\nabla\mathcal{L}(\boldsymbol{x}_k)\|^2=-\gamma_k\|\nabla\mathcal{L}(\boldsymbol{x}_k)\|^2.\end{aligned}$$

Where $\gamma_k=\alpha_k(1-\frac{\beta}{2}(m+4)\alpha_k)$. We now fix $\alpha_k=\alpha=\frac{1}{\beta(m+4)}$ which implies $\gamma_k=\gamma=\frac{1}{2\beta(m+4)}$. Assuming that $\mathcal{L}$ satisfies $\mu-PL$ condition at $\boldsymbol{x}_k$:

$$\mathbb{E}\{\mathcal{L}(\boldsymbol{x}_{k+1})\mid\boldsymbol{x}_k\}\leq\mathcal{L}(\boldsymbol{x}_k)-2\mu\gamma(\mathcal{L}(\boldsymbol{x}_k)-\mathcal{L}^*).$$

By using the tower rule of expectations and induction we have

$$\mathbb{E}\{\mathcal{L}(\boldsymbol{x}_{k+1}) - \mathcal{L}^*\} \leq (1-2\mu\gamma)^k(\mathcal{L}(\boldsymbol{x}_0) - \mathcal{L}_0^*). \quad (9)$$

Invoking $\gamma = \frac{1}{2\beta(m+4)}$ completes the proof. ■

B. Time varying Optimisation

In what follows, we prove a convergence property of online line-search methods for the objective functions that satisfy the following assumptions.

Assumption 13: (Polyak-Łojasiewicz Condition) There exists a positive scalar μ such that $\mathcal{L}_k(\boldsymbol{x})$ satisfies (6) for all k and for all $\boldsymbol{x} \in \mathbb{R}^m$.

We also need to quantify the speed with which the objective function varies in each step. As proved in [6], for having meaningful bounds in nonconvex online optimisation problems, one needs to restrict the changes of the loss functions. To capture this, we assume the following standard assumptions in the literature [10, 14].

Assumption 14: (Drift in Time): There exist non-negative scalars η_0 and η^* such that $\mathcal{L}_{k+1}(\boldsymbol{x}) - \mathcal{L}_k(\boldsymbol{x}) \leq \eta_0$ for all $\boldsymbol{x} \in \mathbb{R}^m$ and $\mathcal{L}_{k+1}^* - \mathcal{L}_k^* \leq \eta^*$ for all k.

Remark 15: In an over-parameterised setting with a nonlinear least square loss function, one can argue that $\mathcal{L}_k^* \approx 0$, for all k. Therefore, η^* is negligible in this setting.

1) Forward gradient convergence in time-varying setting: In this part, we analyse the performance of the forward-gradient algorithm with a fixed step size for the time-varying nonconvex cost functions under PL and smoothness assumptions. We prove that the tracking error linearly converges to a neighbourhood of zero where the rate of convergence depends on the dimension of the problem and the asymptotic error is independent of the fact that we are not using exact gradients.

Theorem 16: Assume that $\mathcal{L}_k$ is $\beta-$smooth for each k, and Assumptions 13 and 14 hold. If we use the directional derivative algorithm with a constant step-size $\alpha_k = \alpha < \frac{2}{\beta(m+4)}$ and random direction U_k is chosen from a standard normal distribution, then

$$\mathbb{E}\{\|\boldsymbol{x}_{k+1} - \pi_{\mathcal{X}_{k+1}^*}(\boldsymbol{x}_{k+1})\|^2\} \leq \frac{\eta}{\mu^2\gamma} + \frac{2}{\mu}(1-2\mu\gamma)^k(\mathcal{L}_0(\boldsymbol{x}_0) - \mathcal{L}_0^*), \quad (10)$$

where $\eta = \eta_0 + \eta^*$ and η_0 and η^* are as in Assumption 14, $\gamma \in (0, \tilde{\gamma})$ where $\tilde{\gamma} = min\{\alpha(1-\frac{\beta}{2}(m+4)\alpha), \frac{1}{2\mu}\}$, and $\mathcal{X}_k^*$ is the set of minimisers of $\mathcal{L}_k(\boldsymbol{x})$.

Proof: Using descent lemma and (7), we have

$$\begin{aligned}
&\mathcal{L}_k(\boldsymbol{x}_{k+1}) - \mathcal{L}_k(\boldsymbol{x}_k) \\
&\leq \langle \nabla\mathcal{L}_k(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k\rangle + \frac{\beta}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2 \\
&= -\alpha_k\langle \nabla\mathcal{L}_k(\boldsymbol{x}_k), \boldsymbol{v}_k(\boldsymbol{x}_k, U_k)\rangle + \frac{\beta}{2}\alpha_k^2\|\boldsymbol{v}_k(\boldsymbol{x}_k, U_k\|^2.
\end{aligned}$$

By taking conditional expectation given $\boldsymbol{x}_k$ with respect to U_k, and using (8) we obtain

$$\begin{aligned}
\mathbb{E}\{\mathcal{L}_k(\boldsymbol{x}_{k+1}) \mid \boldsymbol{x}_k\} &\leq \mathcal{L}_k(\boldsymbol{x}_k) - \alpha_k\|\nabla\mathcal{L}_k(\boldsymbol{x}_k)\|^2 \\
&\quad + \frac{1}{2}\beta\alpha_k^2(m+4)\|\nabla\mathcal{L}_k(\boldsymbol{x}_k)\|^2 \\
&= \mathcal{L}_k(\boldsymbol{x}_k) - \gamma_k\|\nabla\mathcal{L}_k(\boldsymbol{x}_k)\|^2. \quad (11)
\end{aligned}$$

where $\gamma_k = \alpha_k(1 - \frac{\beta}{2}(m+4)\alpha_k)$. We now fix $\epsilon_1 < \alpha_k = \alpha < \frac{2}{\beta(m+4)} - \epsilon_2$ and $\gamma_k = \gamma > 0$ for some $\epsilon_1 > 0$ and $\epsilon_2 > 0$. Assuming that $\mathcal{L}_k$ satisfies $\mu - PL$ condition at $\boldsymbol{x}_k$, for all k, we have that

$$\mathbb{E}\{\mathcal{L}_k(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k\} \leq \mathcal{L}_k(\boldsymbol{x}_k) - 2\mu\gamma(\mathcal{L}_k(\boldsymbol{x}_k) - \mathcal{L}_k^*). \quad (12)$$

By adding and subtracting terms, we have

$$\begin{aligned}
&\mathbb{E}\{\mathcal{L}_{k+1}(\boldsymbol{x}_{k+1}) - \mathcal{L}_{k+1}^*|\boldsymbol{x}_k\} \\
&\quad = \mathbb{E}\{\mathcal{L}_{k+1}(\boldsymbol{x}_{k+1}) - \mathcal{L}_k(\boldsymbol{x}_{k+1})|\boldsymbol{x}_k\} \\
&\quad + \mathbb{E}\{\mathcal{L}_k(\boldsymbol{x}_{k+1}) - \mathcal{L}_k(\boldsymbol{x}_k)|\boldsymbol{x}_k\} \\
&\quad + (\mathcal{L}_k(\boldsymbol{x}_k) - \mathcal{L}_k^*) + (\mathcal{L}_k^* - \mathcal{L}_{k+1}^*) \\
&\quad \overset{(b)}{\leq} \eta + (1-2\gamma\mu)(\mathcal{L}_k(\boldsymbol{x}_k) - \mathcal{L}_k^*),
\end{aligned}$$

where in (b), we have used the bounds on drift and (12). By using the tower rule of expectations and induction, we have

$$\begin{aligned}
\mathbb{E}\{\mathcal{L}_k(\boldsymbol{x}_{k+1}) - \mathcal{L}_{k+1}^*\} &\leq \eta\sum_{i=0}^{k}(1-2\gamma\mu)^{k-i} \\
&\quad + (1-2\mu\gamma)^k(\mathcal{L}_0(\boldsymbol{x}_0) - \mathcal{L}_0^*) \\
&\leq \frac{\eta}{2\mu\gamma} + (1-2\mu\gamma)^k(\mathcal{L}_0(\boldsymbol{x}_0) - \mathcal{L}_0^*).
\end{aligned}$$

Invoking (4) and $\gamma \in (0, \tilde{\gamma})$ completes the proof. ■

Remark 17: In the case of a pre-trained neural network where the term $\mathcal{L}_0(\boldsymbol{x}_0) - \mathcal{L}_0^*$ is small (or zero), the slower convergence rate in higher dimensions is not a problem while the asymptotic error bound, the first term in 10, is of interest.

Remark 18: Under the hypotheses of Theorem 16 the following holds:

$$\limsup_{k\to\infty} \mathbb{E}\{\mathcal{L}_k(\boldsymbol{x}_{k+1}) - \mathcal{L}_{k+1}^*\} \leq \frac{\eta}{2\mu\gamma}. \quad (13)$$

This, in turn, results in $\limsup_{k\to\infty} \|\boldsymbol{x}_k - \pi_{\mathcal{X}_k^*}(\boldsymbol{x}_k)\|^2 \leq \frac{\eta_0+\eta^*}{\mu^2\gamma}$.

Remark 19: One can choose $\alpha_k = \alpha = \frac{1}{\beta(m+4)}$ to maximise the convergence rate, but this results in maximising the asymptotic optimality gap as well.

V. Illustrative Numerical Results

The theoretical error bounds of previous sections are illustrated here employing two simple numerical examples; online least square problem and real-time neural network training.

A. Online Least Square Problem

We consider a sequence of time-varying linear least square problems of the form $\mathcal{L}_k(\boldsymbol{x}) = \frac{1}{2}\|A_k\boldsymbol{x} - \boldsymbol{b}_k\|^2$, where $A_k \in \mathbb{R}^{n\times m}$ and $b_k \in \mathbb{R}^n$. The cost function $\mathcal{L}_k$ satisfies the PL condition for every choice of A_k and $\boldsymbol{b}_k$. We consider the over-parameterised case $m = 60$ and $n = 10$. The vector $\boldsymbol{b}_k$ is generated as $\boldsymbol{b}_{k+1} = \boldsymbol{b}_k + \delta b_k$, where δb_k follows a normal distribution $\mathcal{N}(\boldsymbol{0}, 10^{-2}I_n)$. The matrix A_k is generated using

its singular value decomposition, $A_k = U\Sigma_k V^T$ where $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{m\times r}$ are orthogonal matrices and $\Sigma_{k+1} = \Sigma_k - 10^{-6} I_r$ that $\Sigma_0 = \text{diag}\ \{0.1, 0.2, \ldots, 1\}$ and $r = 10$. This setting assures that $\sup_{\boldsymbol{x}}\{\mathcal{L}_{k+1}(\boldsymbol{x}) - \mathcal{L}_k(\boldsymbol{x})\}$ is bounded. We have run 50 different experiments with the same starting point and have calculated the average loss. Results are plotted in Figure 1, showing the performance of the actual algorithm and the derived theoretical bounds, validating the convergence results.

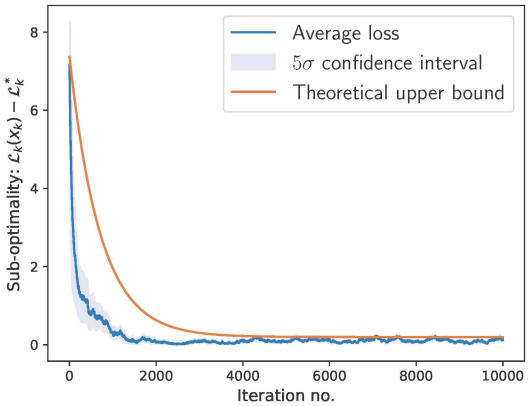

Fig. 1: Performance of online gradient descent algorithm using directional derivatives and the theoretical bound

B. Real Time Neural Network Training

Assume that, while deploying the pre-trained neural network, new data samples arrive at a high rate and one needs to update the training variable of the network, $\boldsymbol{x}$. In the online setting, the neural net receives an input $\boldsymbol{a}_i$ at each round i, and with $\boldsymbol{x}$ suffers loss $\frac{1}{2}\|f(\boldsymbol{x};\boldsymbol{a}_i) - b_i\|^2$ where $f(\boldsymbol{x};\boldsymbol{a})$ is deep neural network's output for input $\boldsymbol{a}$ and parameter $\boldsymbol{x}$. Note that this framework generalises the supervised learning paradigm. The corresponding cost functions are

$$\mathcal{L}_k(\boldsymbol{x}) = \frac{1}{2}\sum_{i=k}^{k+n-1} \|f(\boldsymbol{x};\boldsymbol{a}_i) - b_i\|^2.$$

We assume that the algorithm uses the latest n samples (because they represent the data distribution better) and due to its fast computation, the forward gradient is to be used in the optimisation steps.

Consider a scenario where while deploying a pre-trained neural network, the model observes new data samples with a different distribution than the training data. To demonstrate this, we consider a simple two-layer neural network trained on MNIST data set [18]. The neural net has two hidden dense layers with 32 and 16 nodes, respectively. We trained the model on the first 30000 samples with a 92.6% test accuracy. By changing the distribution of the other 40000 samples (see 2, we observe that the performance decreases to 53% accuracy. We assume the model observes 30,000 noisy samples one by one and updates its weights after observing every new sample. We set to use another 10,000 noisy samples as a test data set and measure the instantaneous performance of the evolving model on the unseen samples for both the online gradient method and the proposed forward gradient method.

Fig. 2: Noisy samples and the original data

Figure 3 demonstrated the performance of the evolving model on the unseen data for two different methods, gradient-based algorithm and forward gradient method with different learning rates. One can observe that the performance of the forward gradient method is following that of the gradient method asymptotically, validating the theoretical results.

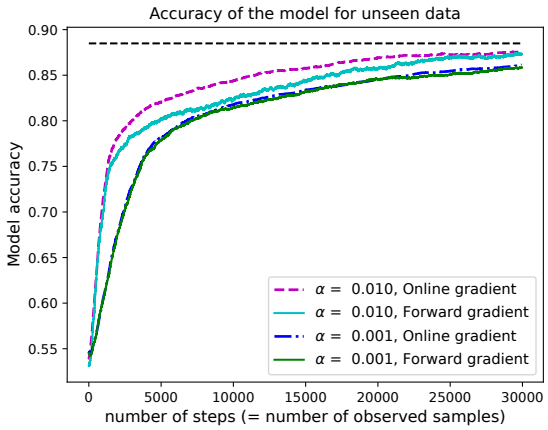

Fig. 3: Performance of the online forward gradient method compared to online online gradient method

VI. Conclusion

In this paper, we analysed the performance of gradient-based first-order algorithms focusing on faster algorithms for calculating gradients, namely forward gradients which are particularly useful in computation and memory-limited applications. We have exploited the fact that non-linear least square problems in over-parameterised settings will satisfy the PL condition. Based on this observation, we proved that the (proximal-)gradient-based algorithm based on the forward mode of automatic differentiation (forward gradient) with nonconvex objective functions converges to optimal value function at a linear rate. We have also analysed the convergence of these algorithms in a time-varying setting and proved the linear convergence to the neighbourhood of a global minimiser. This paper gives new insights into using the forward mode of automatic differentiation in problems with limited resources and fast-changing cost functions where calculating full gradients using the backpropagation algorithm is either impossible or inefficient.

REFERENCES

[1] Naman Agarwal, Alon Gonen, and Elad Hazan. "Learning in non-convex games with an optimization oracle". In: *Conference on Learning Theory*. PMLR. 2019, pp. 18–29.

[2] Atilim Gunes Baydin et al. "Automatic differentiation in machine learning: a survey". In: *Journal of Marchine Learning Research* 18 (2018), pp. 1–43.

[3] Atılım Güneş Baydin et al. "Gradients without Backpropagation". In: *arXiv preprint arXiv:2202.08587* (2022).

[4] Mikhail Belkin. "Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation". In: *Acta Numerica* 30 (2021), pp. 203–248.

[5] Dimitri P Bertsekas. "Nonlinear programming". In: *Journal of the Operational Research Society* 48.3 (1997), pp. 334–334.

[6] Omar Besbes, Yonatan Gur, and Assaf Zeevi. "Non-stationary stochastic optimization". In: *Operations research* 63.5 (2015), pp. 1227–1244.

[7] Xuanyu Cao, Junshan Zhang, and H Vincent Poor. "Online stochastic optimization with time-varying distributions". In: *IEEE Transactions on Automatic Control* 66.4 (2020), pp. 1840–1847.

[8] Olivier Devolder, François Glineur, and Yurii Nesterov. "First-order methods of smooth convex optimization with inexact oracle". In: *Mathematical Programming* 146.1 (2014), pp. 37–75.

[9] William Fedus, Barret Zoph, and Noam Shazeer. *Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity*. 2021.

[10] Xiand Gao, Xiaobo Li, and Shuzhong Zhang. "Online learning with non-convex losses and non-stationary regret". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 235–243.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[12] Elad Hazan et al. "Introduction to online convex optimization". In: *Foundations and Trends® in Optimization* 2.3-4 (2016), pp. 157–325.

[13] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems* 31 (2018).

[14] Ali Jadbabaie et al. "Online optimization: Competing with dynamic comparators". In: *Artificial Intelligence and Statistics*. PMLR. 2015, pp. 398–406.

[15] Hamed Karimi, Julie Nutini, and Mark Schmidt. "Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2016, pp. 795–811.

[16] Ahmed Khaled and Peter Richtárik. "Better theory for SGD in the nonconvex world". In: *arXiv preprint arXiv:2002.03329* (2020).

[17] Seunghyun Kim, Liam Madden, and Emiliano Dall'Anese. "Convergence of the Inexact Online Gradient and Proximal-Gradient Under the Polyak-{\L} ojasiewicz Condition". In: *arXiv preprint arXiv:2108.03285* (2021).

[18] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[19] Chaoyue Liu, Libin Zhu, and Mikhail Belkin. "Loss landscapes and optimization in over-parameterized non-linear systems and neural networks". In: *Applied and Computational Harmonic Analysis* 59 (2022), pp. 85–116.

[20] Chaoyue Liu, Libin Zhu, and Misha Belkin. "On the linearity of large non-linear models: when and why the tangent kernel is constant". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 15954–15964.

[21] Liam Madden, Stephen Becker, and Emiliano Dall'Anese. "Bounds for the tracking error of first-order online optimization methods". In: *Journal of Optimization Theory and Applications* 189.2 (2021), pp. 437–457.

[22] Yurii Nesterov and Vladimir Spokoiny. "Random gradient-free minimization of convex functions". In: *Foundations of Computational Mathematics* 17.2 (2017), pp. 527–566.

[23] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[24] Ana M Ospina, Nicola Bastianello, and Emiliano Dall'Anese. "Feedback-Based Optimization With Sub-Weibull Gradient Errors and Intermittent Updates". In: *IEEE Control Systems Letters* 6 (2022), pp. 2521–2526.

[25] Boris T Polyak. "Gradient methods for the minimisation of functionals". In: *USSR Computational Mathematics and Mathematical Physics* 3.4 (1963), pp. 864–878.

[26] Mark Schmidt, Nicolas Roux, and Francis Bach. "Convergence rates of inexact proximal-gradient methods for convex optimization". In: *Advances in neural information processing systems* 24 (2011).

[27] Shai Shalev-Shwartz et al. "Online learning and online convex optimization". In: *Foundations and Trends® in Machine Learning* 4.2 (2012), pp. 107–194.

[28] Iman Shames and Farhad Farokhi. "Online stochastic convex optimization: Wasserstein distance variation". In: *arXiv preprint arXiv:2006.01397* (2020).

[29] Arun Sai Suggala and Praneeth Netrapalli. "Online non-convex learning: Following the perturbed leader is optimal". In: *Algorithmic Learning Theory*. PMLR. 2020, pp. 845–861.

[30] Sharan Vaswani, Francis Bach, and Mark Schmidt. "Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron". In: *The 22nd international conference on artificial intelligence and statistics*. PMLR. 2019, pp. 1195–1204.

[31] Silvia Villa et al. "Accelerated and inexact forward-backward algorithms". In: *SIAM Journal on Optimization* 23.3 (2013), pp. 1607–1633.

[32] Tianbao Yang et al. "Tracking slowly moving clairvoyant: Optimal dynamic regret of online learning with true and noisy gradient". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 449–457.

[33] Martin Zinkevich. "Online convex programming and generalized infinitesimal gradient ascent". In: *Proceedings of the 20th international conference on machine learning (icml-03)*. 2003, pp. 928–936.