2022 IEEE 61st Conference on Decision and Control (CDC)
December 6-9, 2022. Cancún, Mexico

One-Pass Learning via Bridging Orthogonal Gradient Descent and Recursive Least-Squares

Youngjae Min, Kwangjun Ahn, and Navid Azizan

2022 IEEE 61st Conference on Decision and Control (CDC) | 978-1-6654-6761-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CDC51059.2022.9992939

***Abstract*— While deep neural networks are capable of achieving state-of-the-art performance in various domains, their training typically requires iterating for many passes over the dataset. However, due to computational and memory constraints and potential privacy concerns, storing and accessing all the data is impractical in many real-world scenarios where the data arrives in a stream. In this paper, we investigate the problem of *one-pass learning*, in which a model is trained on sequentially arriving data without retraining on previous datapoints. Motivated by the increasing use of overparameterized models, we develop *Orthogonal Recursive Fitting* (ORFit), an algorithm for one-pass learning which seeks to perfectly fit every new datapoint while changing the parameters in a direction that causes the least change to the predictions on previous datapoints. By doing so, we bridge two seemingly distinct algorithms in adaptive filtering and machine learning, namely the *recursive least-squares* (RLS) algorithm and *orthogonal gradient descent* (OGD). Our algorithm uses the memory efficiently by exploiting the structure of the streaming data via an incremental principal component analysis (IPCA). Further, we show that, for overparameterized linear models, the parameter vector obtained by our algorithm is what stochastic gradient descent (SGD) would converge to in the standard multi-pass setting. Finally, we generalize the results to the nonlinear setting for highly overparameterized models, relevant for deep learning. Our experiments show the effectiveness of the proposed method compared to the baselines.**

I. Introduction

While deep neural networks have been successful in numerous domains, their training is computationally demanding and requires iterating over the entire dataset multiple times. This hinders their deployment in many real-world settings such as robot learning and online decision making, where new datapoints are collected over time or become available sequentially. In such settings, storing all the datapoints and retraining the model at every step on all the data is extremely costly and often not feasible. In addition, certain applications may prohibit storing the data for privacy reasons.

Thus, it is very desirable to come up with algorithms that can learn incrementally or in an online fashion, rather than by iterating over the entire data many times. However, it is well-known that neural networks are prone to "catastrophically forget" past information while learning new data [9]. This begs the question:

"Can we learn streaming data efficiently without forgetting or retraining on previous data?"

This is a setting often referred to as *one-pass learning*. More specifically, one-pass learning concerns the setting where the algorithm (i) makes an update for the current datapoint without direct access to previous data; (ii) the new updates do not significantly affect the predictions on the previous data; moreover, (iii) the computational and memory costs of each update must not grow with the iteration count.

There has been growing attention on one-pass learning and its variants. In particular, [7] studied learning the ImageNet dataset in a single pass by revisiting some "important" previous datapoints at each learning step, and [18] investigated learning incremental 'batches' on a large scale by correcting the classifier's bias towards new data. However, both methods train on previous data and are not adequate for one-pass learning. On the other hand, [14] proposed an effective one-pass learning method for support vector machines. However, their method is tailored to the specific setting of support vector machines. Further, [15] proposed a one-pass deep learning algorithm, but it relies on a specific network architecture and is vulnerable to forgetting the previous data unless the data is consistently arriving from the same distribution.

One-pass learning is also closely related to a classical problem studied in the context of control/estimation theory. More specifically, a classical algorithm known as recursive least-squares (RLS) (see, *e.g.*, [16]) tackles one-pass learning for linear models (as elaborated in Section II-B). However, there are two limitations of the standard RLS: (i) it suffers from high computational and memory costs, and (ii) it is not well-suited for the overparameterized setting where zero training loss is desired. The main focus of this work is to develop a method that overcomes these limitations. Related to the overparameterized setting, a few works have recently discussed utilizing RLS to train popular deep neural networks [19], [20]. However, these works are empirical in nature and consider multi-pass learning with mini-batches. Moreover, the theoretical properties of RLS for one-pass learning are not studied in those works.

A. Contributions

Our main contributions can be summarized as follows.

- We develop Orthogonal Recursive Fitting (ORFit), an algorithm for one-pass learning in the overparameterized setting which fits new data on the fly while updating the parameters in a direction that causes the least change to the predictions on previous data. Our algorithm uses memory efficiently by exploiting the structure of the streaming data via incremental principal component analysis (IPCA) to extract the essential information for the update. (§III)

The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. {yjm,kjahn,azizan}@mit.edu

- Through the proposed method, we establish an interesting connection between two different algorithms from adaptive filtering and machine learning, namely, the recursive least-squares (RLS) algorithm and the orthogonal gradient descent (OGD). We further theoretically characterize the behavior of the proposed method in the linear overparameterized setting. (§IV)
- We demonstrate the practicality of our approach and corroborate our theoretical findings through various experiments. (§V)
- We discuss further extensions to overparameterized nonlinear models, relevant for deep learning. (§VI)

II. Preliminaries

A. One-Pass Learning

Let $f(x;w) \in \mathbb{R}$ be a model that the agent is trying to fit, where $x \in \mathscr{X} \subset \mathbb{R}^d$ is the input and $w \in \mathbb{R}^p$ is the parameter (weight) vector. Consider a sequentially arriving stream of data $\{(x_k, y_k)\}_{k=1}^K$ where $x_k \in \mathscr{X}$ and $y_k \in \mathscr{Y} \subset \mathbb{R}$. In overparameterized models, we have $p \geq K$ (and often $p \gg K$). For a loss function $\ell(\cdot,\cdot)$, let $f_k(w) := f(x_k;w)$ and $\ell_k(w) := \ell(y_k, f_k(w))$. Then, one-pass learning considers the setting where, given an initial parameter $w_0 \in \mathbb{R}^p$, it updates the parameter $w_i \in \mathbb{R}^p$ after the new data (x_i, y_i) arrived without revisiting previous datapoints $\{(x_k, y_k)\}_{k=1}^{i-1}$.

For concreteness, we first focus on a linear overparameterized model $f(x;w) = w^\top x$ with $p = d$. We then discuss the extension of the results to nonlinear models (*e.g.* overparameterized architectures in deep learning) in Section VI.

Before introducing our algorithm and the results, we briefly review two related algorithms capable of one-pass learning, proposed in two different literatures.

B. Recursive Least-Squares

First, we briefly review recursive least-squares (RLS) from the control/estimation theory literature (refer to [16] for details). At every step i, RLS aims to find a parameter vector that solves the following regularized least-squares problem:

$$w_i^{(RLS)} = \arg\min_w \sum_{k=1}^{i} (y_k - w^\top x_k)^2 + \|w - w_0\|_\Pi^2, \quad (1)$$

where $\|x\|_\Pi := \sqrt{x^\top \Pi x}$ for a $p \times p$ positive-definite matrix Π and w_0 is an initial parameter estimate. Note that the system is underdetermined/overparameterized, and the regularization term in (1) is necessary for the solution to be uniquely defined. While there is a closed-form solution for (1) given by $w_i^{(RLS)} = (\Pi + X_i^\top X_i)^{-1}(X_i^\top Y_i + \Pi w_0)$ with $X_i = [x_1\ x_2 \dots\ x_i]^\top$ and $Y_i = [y_1\ y_2 \dots\ y_i]^\top$, computing the solution directly requires storing all the previous data as well as recomputing the inverse operation for every new datapoint. RLS bypasses this issue by computing the new solution $w_i^{(RLS)}$ of (1) recursively from $w_{i-1}^{(RLS)}$ and (x_i, y_i).

We elaborate the algorithm more formally for a general version of RLS called *exponentially-weighted recursive least-squares (EW-RLS)* [16]. Consider the following problem:

$$w_i^{(RLS)} = \arg\min_w \sum_{k=1}^{i} \lambda^{i-k}(y_k - w^\top x_k)^2 + \lambda^i \|w - w_0\|_\Pi^2, \quad (2)$$

with a forgetting factor $0 < \lambda \leq 1$. Note that this reduces to the problem of the vanilla RLS (1) when $\lambda = 1$. The exact solution of it is recursively updated as follows:

$$\begin{cases} w_i^{(RLS)} = w_{i-1}^{(RLS)} + \dfrac{P_{i-1}x_i}{\lambda^i + x_i^\top P_{i-1} x_i}(y_i - x_i^\top w_{i-1}^{(RLS)}), \\ P_i = P_{i-1} - \dfrac{P_{i-1}x_i x_i^\top P_{i-1}}{\lambda^i + x_i^\top P_{i-1} x_i}, \end{cases} \quad (3)$$

with $w_0^{(RLS)} = w_0$ and $P_0 = \Pi^{-1}$. Here, P_i can be alternatively written as $P_i = [\Pi + X_i^\top \Lambda_i X_i]^{-1}$ where $\Lambda_i = \mathrm{diag}(\lambda^{-1}, \lambda^{-2}, \dots, \lambda^{-i})$.

C. Orthogonal Gradient Descent

An algorithm called orthogonal gradient descent (OGD) [6] has been recently proposed in the context of *continual learning*. More specifically, OGD considers sequentially arriving tasks $\{T_1, T_2, \dots\}$, where each task consists of a set of datapoints. Continual learning can be understood as a "batch" version of one-pass learning. At a high level, when the i-th task T_i arrives, OGD updates the parameter to fit new samples from T_i in a way that causes minimal changes to the predictions for previous tasks $\{T_k\}_{k=1}^{i-1}$. The gradient of the model on a datapoint x_j with respect to the parameter, $\nabla_w f(x_j;w)$, is the direction in the parameter space that causes the most change to the prediction on that datapoint. Thus, moving orthogonal to this direction, locally, keeps the prediction unchanged, which is the main idea behind OGD. More formally, the update direction is computed via projecting the current gradient g of the loss onto the subspace orthogonal to $\mathscr{G} := \mathrm{span}\{\bigcup_{k=1}^{i-1}\{\nabla_w f(x;w_k)\}_{(x,y)\in T_k}\}$:

$$\tilde{g} = g - \sum_{v \in S} \mathrm{proj}_v(g), \quad (4)$$

where S is an orthogonal basis for $\mathscr{G}$ and $\mathrm{proj}_v(u) := (u^\top v/\|v\|^2)v = (vv^\top/\|v\|^2)u$. The orthogonal basis S is incrementally updated through the Gram-Schmidt procedure.

III. Orthogonal Recursive Fitting

In this section, we propose a one-pass learning algorithm called *orthogonal recursive fitting* (ORFit). The algorithm consists of three main components: (i) orthogonal update of the parameter motivated by OGD; (ii) interpolation (perfect fitting) of new data in a single step; and (iii) efficient use of memory via incremental summary. We describe the details of each component below, starting from (i) and (ii). See Fig. 1 for an illustration.

A. Orthogonal Recursive Update

We start by considering OGD directly applied to the one-pass learning setting. By treating each task T_k in the continual learning setting as consisting of a single datapoint, OGD will run multiple gradient descent steps on the single datapoint. While perfect fitting/interpolation is desired in often highly overparameterized models [4], [5], it will take many iterations for OGD to perfectly fit the datapoint. Instead, inspired by the recent trend in meta-learning, we

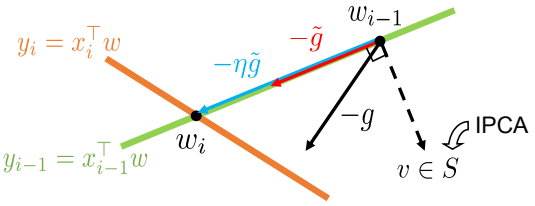


Fig. 1: An illustration of ORFit in the parameter space for a linear model. The parameter w_{i-1} fits the previous datapoints $\{(x_k, y_k)\}_{k=1}^{i-1}$. The set S consists of the directions moving towards which causes the most change in the predictions on previous data, and thus, moving orthogonal to S keeps the predictions intact. Given a new datapoint (x_i, y_i), projecting its corresponding gradient g to the orthogonal complement of the subspace spanned by S yields the new update direction $\tilde{g}$. ORFit finds a new parameter w_i along the direction of $-\tilde{g}$ which fits the new datapoint (x_i, y_i) within a single step, while still fitting the previous data $\{(x_k, y_k)\}_{k=1}^{i-1}$.

consider a one-step learning scheme that only runs a *single* gradient step to interpolate the new datapoint. We first begin with the following result that serves as a building block for our algorithm design; see our full report [13] for a proof.

Lemma 1. *Consider a linear model $f(x;w) = w^\top x$, and let $\tilde{g}$ be the projection defined by (4) of any vector $g \in \mathbb{R}^p$. Then, for any step size $\eta \in \mathbb{R}$, the new parameter $w' = w - \eta\tilde{g}$ preserves the predictions on the previous datapoints, i.e., $f(x;w') = f(x;w)$ for all $(x,y) \in \bigcup_{k=1}^{i-1} T_k$.*

The main takeaway of Lemma 1 is that the predictions for the previous datapoints do not change when we update the model along the direction $\tilde{g}$. Hence, we may choose η so that the updated parameter w' can perfectly fit the new datapoint, say (x', y'), as $y' = w'^\top x' = (w - \eta\tilde{g})^\top x'$. Following this principle, a straightforward calculation yields the following update rule for $i \geq 1$ with $S_0 = \phi$ and the initial weight w_0:

$$\begin{cases} \tilde{g}_{i-1} = \nabla \ell_i(w_{i-1}) - \sum_{v \in S_{i-1}} \mathrm{proj}_v(\nabla \ell_i(w_{i-1})), \\ S_i = S_{i-1} \bigcup \{\nabla f_i(w_{i-1}) - \sum_{v \in S_{i-1}} \mathrm{proj}_v(\nabla f_i(w_{i-1}))\}, \\ w_i = w_{i-1} - \eta_{i-1}\tilde{g}_{i-1}, \end{cases} \tag{5}$$

where the optimal step size is chosen as

$$\eta_{i-1} = \frac{1}{\nabla f_i(w_{i-1})^\top \tilde{g}_{i-1}} (f_i(w_{i-1}) - y_i). \tag{6}$$

For intuition, we note that the optimal step size (6) is typically small for highly overparameterized models; there are many parameter vectors in the vicinity of the current solution that perfectly fit the new datapoint [1], [2], [12].

Remark 1 (Computational overhead). It is important to note that all the quantities appearing in (6) are typically available in the gradient-based optimization setting, and hence, there is no computational overhead for computing the stepsize (6).

Remark 2 (Nonlinear models). Although the update rule (5) is derived based on linear models, it can also be applied to highly overparameterized *nonlinear* models such as deep neural networks, as we will discuss in Section VI.

Another distinction between the update rule (5) and OGD lies in the update of the orthogonal basis S_i: OGD utilizes "fresher" gradient at the updated parameter w_i by $S_i = S_{i-1} \bigcup \{\nabla f_i(w_i) - \sum_{v \in S_{i-1}} \mathrm{proj}_v(\nabla f_i(w_i))\}$. Although the two bases actually span the same subspace for linear models, it turns out that ORFit in (5) leads to a natural generalization for nonlinear models; see Section VI for details.

Although the update rule (5) does not access previous datapoints, it still requires storing the orthogonal basis S_i, whose size grows linearly in the number of visited datapoints. This is not desirable in practice when one needs to train the model on a large dataset. We address this issue next.

B. Incremental Summary of Memory

In this section, we overcome the aforementioned memory issue by utilizing the structure of the streaming dataset. The main idea is to summarize the orthogonal basis S using an incremental principal component analysis (IPCA) algorithm, known as the sequential Karhunen–Loeve (SKL) algorithm proposed in [11]. IPCA is a memory-efficient variant of PCA that enables sequential update for streaming/large datasets. Let us formally describe how IPCA incrementally summarizes the orthogonal basis.

Consider the orthogonal basis $S = \{v_1, v_2, \ldots, v_i\}$ obtained by (5). Let the singular value decomposition (SVD) of $A = [v_1\ v_2 \ldots v_i]$ be $A = U\Sigma V^\top$. Here, the crucial information of the SVD is the left-singular vectors $\mathrm{col}(U)$ that forms an orthonormal basis for $\mathrm{span}(S)$. This information can be used to come up with a rank-m approximation of A by using the components corresponding to the top m singular values.

Now suppose that the orthogonal basis S is augmented with a newly projected gradient v' as in (5). We want to efficiently update U for the new basis. Letting $u := v'/\|v'\|$, the new basis matrix can be represented as:

$$\begin{bmatrix} A & v' \end{bmatrix} = \begin{bmatrix} U & u \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & u^\top v' \end{bmatrix} \begin{bmatrix} V^\top & 0 \\ 0 & 1 \end{bmatrix} \tag{7}$$

$$= \left(\begin{bmatrix} U & u \end{bmatrix} \tilde{U} \right) \tilde{\Sigma} \left(\tilde{V}^\top \begin{bmatrix} V^\top & 0 \\ 0 & 1 \end{bmatrix} \right), \tag{8}$$

where $\tilde{U}\tilde{\Sigma}\tilde{V}^\top$ is the SVD of $\begin{bmatrix} \Sigma & 0 \\ 0 & u^\top v' \end{bmatrix}$. Then, (8) is the SVD of the new basis matrix. Hence to update U, one can directly use the information from the previous iteration, namely U and Σ. The important aspect here is that the update can be made *without storing* V and *without having to recompute* the SVD of the new basis matrix. Finally, one can store only the top m singular values in Σ and their corresponding components in U. By repeatedly applying this IPCA algorithm in addition to (5), we obtain ORFit. See Algorithm 1 for the detailed procedure.

The main advantage of ORFit is its **computational/memory efficiency**. By only storing the top m components, we can reduce the memory size from $O(ip)$ to $O(mp)$. Moverover, the additional computational overhead to perform IPCA as well as the total time complexity of ORFit at each step is $O(m^2(p+m))$. Hence, ORFit can reduce both the computation and the memory complexity with a proper m.

Algorithm 1 Orthogonal Recursive Fitting (ORFit)

Input: Data sequence $((x_k, y_k))_{k=1}^K$, memory limit m
Output: The optimal parameter w

Initialize $U \leftarrow [\ ]$, $\Sigma \leftarrow [\ ]$, $w \leftarrow w_0$
for $i = 1, 2, 3, \ldots$ **do**
  $g \leftarrow$ Stochastic Gradient for (x_i, y_i) at w
  $\tilde{g} \leftarrow g - \sum_{v \in \mathrm{col}(U)} \mathrm{proj}_v(g)$
  $v' \leftarrow \nabla f_i(w) - \sum_{v \in \mathrm{col}(U)} \mathrm{proj}_v(\nabla f_i(w))$
  if $i \leq m$ **then**
    $U \leftarrow [U\ v']$
    if $i = m$ **then**
      $U, \Sigma \leftarrow$ Compute SVD of U
    end if
  else
    $u \leftarrow v'/\|v'\|$
    $\tilde{U}, \Sigma \leftarrow$ Compute SVD of $\begin{bmatrix} \Sigma & 0 \\ 0 & u^\top v' \end{bmatrix}$
    $U \leftarrow [U\ u]\tilde{U}$
    $U, \Sigma \leftarrow$ top m singular vectors/values in U, Σ
  end if
  $\eta \leftarrow (f_i(w) - y_i)/(\nabla f_i(w)^\top \tilde{g})$
  $w \leftarrow w - \eta \tilde{g}$
end for

IV. Theoretical Results

In this section, we provide the theoretical properties of the proposed method. We begin by discussing a formal connection between the proposed method and RLS.

A. Connection to RLS

It turns out ORFit corresponds to an extreme case of the well-known RLS method, as formally described in the following result; see our full report [13] for a proof.

Proposition 2. *Consider a linear overparameterized ($p \geq K$) model. Let w_0 be the initialization and m be the memory limit for* ORFit*. Then, at each iteration $i \leq m$, the update rule of* ORFit *results in the same parameter vector as the EW-RLS update rule (3) does with $\lambda = 0$, $\Pi = I$, and initialization w_0. In this setting, P_i in (3) is the projection matrix onto the subspace orthogonal to* $\mathrm{span}\{\nabla f_k(w_{k-1})\}_{k=1}^i$.

Remark 3. The optimization problem of EW-RLS (2) is not well-defined for $\lambda = 0$. That said, the update rule (3) can be still computed for $\lambda = 0$, and ORFit finds the same solution as the limiting case of EW-RLS.

Remark 4. ORFit in (5) has $O(ip)$ time and memory complexities, compared to those of $O(p^2)$ for the EW-RLS update rule, where typically $i \ll p$ in the overparameterized setting.

One notable aspect of ORFit is that it bridges the two seemingly distinct algorithms OGD and RLS through Proposition 2. The connection provides us new insights into understanding the behavior of ORFit, as we discuss next.

B. Characterizing the Solution of ORFit

Before presenting our main result, we first provide some intuitions. To understand the behavior of ORFit, let us first recall that the EW-RLS update rule (3) is the solution to the optimization problem (2). In light of Proposition 2, one might be tempted to claim that ORFit in (5) solves (2) with $\lambda = 0$ and $\Pi = I$. However, (2) is not well-defined for $\lambda = 0$.

Nevertheless, intuitively one can regard ORFit as solving (2) in the limit of $\lambda \to 0^+$. Then, for sufficiently small $\lambda > 0$, the first term in the objective of (2) outweighs the second term, which suggests that, in the overparameterized case, the solution should enforce $y_k \approx w^\top x_k$ for all $k = 1, 2, \ldots, i$, while minimizing $\|w - w_0\|^2$. In the following theorem, we formalize this intuition and characterize the solution of ORFit; see our full report [13] for a proof.

Theorem 3. *Consider a linear overparameterized ($p \geq K$) model. Let w_0 be the initialization and m be the memory limit. Then, at each iteration $i \leq m$, the parameter vector obtained by* ORFit *is the solution of the following problem:*

$$w_i = \arg\min_w \ \|w - w_o\| \quad \text{s.t. } y_k = w^\top x_k \quad k = 1, 2, \ldots, i. \tag{9}$$

It is known that, for a linear overparameterized model, in the standard multi-pass learning setting over the dataset $\{(x_k, y_k)\}_{k=1}^i$, as the number of iterations goes to infinity, the iterates of stochastic gradient descent (SGD) initialized at w_0 with a sufficiently small step size converge to the solution of problem (9) (see, *e.g.*, Proposition 1 in [3]). Thus, ORFit with just an epoch of training finds the solution that SGD in the limit of infinite number of iterations converges to.

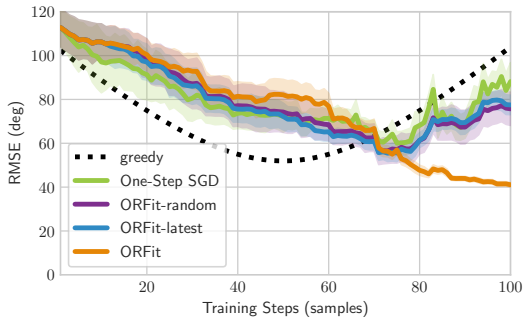
Corollary 4. *Consider a linear overparameterized ($p \geq K$) model. Let w_0 be the initialization and m be the memory limit. The parameter vector obtained by* ORFit *at each iteration $i \leq m$ is equal to what SGD would converge to by iterating over the dataset $\{(x_k, y_k)\}_{k=1}^i$ with a sufficiently small step size in the limit of infinite number of iterations.*

V. Experiments

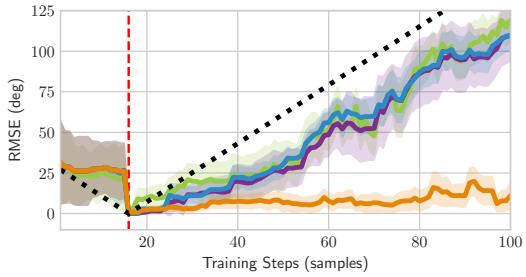
In this section, we demonstrate the effectiveness of our proposed methods in the one-pass learning setting and corroborate the theoretical results presented in Section IV. We performed experiments for linear models in the *Rotated MNIST* setup described in [17]. In this setup, the inputs are rotated MNIST images for digit '2', whose size is 28×28. Our goal is to estimate the rotated angles in $[0, \pi]$. For the training dataset, the angles are uniformly sampled from $[0, \pi]$, and to introduce distribution shift, we order the dataset so that an image rotated with a smaller angle arrives earlier.

A. Learning with Restrictions on Memory Size

In this experiment, we demonstrate the effectiveness of our proposed method in the memory-restricted setting, in which 100 datapoints are sequentially learned while we are only allowed to store up to 10 basis vectors. For comparison, we consider the following baselines: (1) *Greedy scheme*: outputs only the label of the most recently learned datapoint, regardless of the input, as an extreme case of forgetting. (2) *One-Step SGD*: employs the one-step learning scheme with the step size in (6) but with empty orthogonal basis. (3) ORFit-*random*: ORFit that keeps 10 randomly chosen basis



(a) Test Error



(b) Sample Prediction Error

Fig. 2: Results for the memory-restricted setting (§V-A). (a) shows the evolution of the test errors measured after learning each datapoint, while (b) shows the evolution of the prediction errors for a particular sample (the 16-th example) after each iteration. The red dashed line indicates the step on which the sample is trained. The shades indicate the standard deviations over 10 independent runs

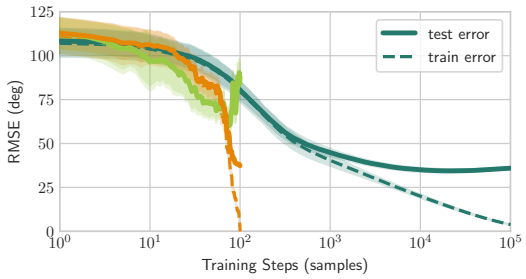
vectors after each iteration instead of performing IPCA. (4) ORFit-*latest*: ORFit that keeps the latest 10 basis vectors after each iteration instead of performing IPCA.

We first compared the test errors of the proposed method and the baselines. The test errors are measured with the test dataset that consists of 1032 images of digit '2' in the MNIST test set of which each is rotated with a random angle in $[0, \pi]$. As shown in Fig. 2a, ORFit outperforms other baselines after a sufficient number of training steps. Notably, ORFit results in lower variances over the 10 independent runs with different initialization.

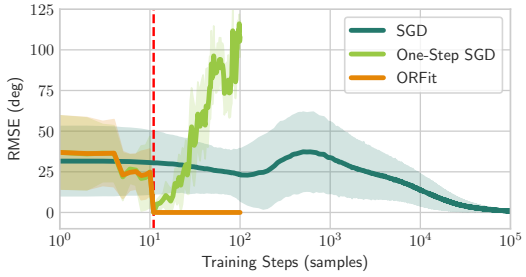
Next, we compared the degrees of forgetting for the proposed method and the baselines by keeping track of the prediction errors of a training datapoint throughout the training. As shown in Fig. 2b, ORFit successfully keeps the prediction error low throughout the training. This is in stark contrast with other methods for which the prediction error quickly increases as other datapoints are learned.

B. Learning without Memory Restriction

In this experiment, we follow the setup in Section V-A except that this time, we do not impose any memory restrictions. For comparison, we run vanilla SGD with a fixed step size 10^{-5}. Vanilla SGD makes multiple passes over the entire dataset for 1000 epochs. Reported in Fig. 3a are the training and test errors of vanilla SGD, One-Step SGD, and ORFit. Note that both SGD and ORFit learn the training



(a) Test & Train Error



(b) Sample Prediction Error

Fig. 3: Results for the setting without memory restriction (§V-B). (a) shows the evolution of the test and train errors measured after each training step, while (b) shows the evolution of the prediction errors for a particular sample (the 11-th example) after each iteration.

dataset perfectly with almost zero training error. Moreover, after the training is finished, SGD and ORFit achieve similar test errors, corroborating Theorem 3. In addition, Fig. 3b shows the prediction error of the 11-th training datapoint throughout the training (analogous to Fig. 2b). Note that ORFit perfectly preserves the prediction error of the sample, while One-Step SGD quickly deteriorates it. This result demonstrates the effectiveness of the orthogonal update in ORFit for one-pass learning.

VI. Extension to Deep Learning

In this section, we extend our discussion to nonlinear models under the neural tangent kernel (NTK) regime [8], [10]. The main idea behind the NTK regime is that a neural network with large enough width is well-approximated by its first-order approximation around the initialization:

$$f_k(w) \approx f_k(w_0) + \nabla f_k(w_0)^\top (w - w_0). \tag{10}$$

In particular, Lee et al. [10] discussed sufficient conditions (in terms of the width of the network) for which this approximation is valid; see their Theorem 2.1 for details. Under the linearized regime, we consider expanding the model around the parameter learned at the previous step as

$$f_k(w) \approx f_k(w_{k-1}) + \nabla f_k(w_{k-1})^\top (w - w_{k-1}). \tag{11}$$

Throughout, we denote by $f_{k|k-1}(w)$ the RHS of (11).

It is notable that ORFit is applicable to any differentiable nonlinear model f. This is in contrast to the EW-RLS algorithm (3), which is only applicable to linear models $f(x; w) = w^\top x$. Based on this observation, we employ the

step size calculated in (6) to fit the new data for a nonlinear model under the NTK regime (11). More specifically, we consider an analog of the EW-RLS (2) for nonlinear models:

$$w_i = \arg\min_{w} \sum_{k=1}^{i} \lambda^{i-k}(y_k - f_{k|k-1}(w))^2 + \lambda^i \|w - w_0\|_{\Pi}^2, \quad (12)$$

given a forgetting factor $0 < \lambda \leq 1$ and $\Pi \succ 0$. Then similarly as in (3), one can write out the solution of (12) in a recursive manner, while treating $(\nabla f_k(w_{k-1}), y_k - f_k(w_{k-1}) + \nabla f_k(w_{k-1})^\top w_{k-1})$ as the streamed data at the k-th step:

$$\begin{cases} w_i = w_{i-1} + \dfrac{P_{i-1}\nabla f_i(w_{i-1})(y_i - f_i(w_{i-1}))}{\lambda^i + \nabla f_i(w_{i-1})^\top P_{i-1}\nabla f_i(w_{i-1})}, \\ P_i = P_{i-1} - \dfrac{P_{i-1}\nabla f_i(w_{i-1})\nabla f_i(w_{i-1})^\top P_{i-1}}{\lambda^i + \nabla f_i(w_{i-1})^\top P_{i-1}\nabla f_i(w_{i-1})}, \end{cases} \quad (13)$$

with initialization w_0 and $P_0 = \Pi^{-1}$. We call this generalized update rule *NTK-RLS*. Similarly as in Section IV, we obtain the following result; see our full report [13] for a proof.

Theorem 5. *Consider a (nonlinear) overparameterized model with $p \geq K$. Let w_0 be the initialization and m be the memory limit for* ORFit. *Then, at each iteration $i \leq m$, the update rule of* ORFit *results in the same parameter vector as the NTK-RLS update rule (13) does with $\lambda = 0$, $\Pi = I$, and initialization w_0. Moreover, at each iteration $i \leq m$, the parameter vector obtained by* ORFit *is the solution of the following optimization problem:*

$$w_i = \arg\min_{w} \|w - w_o\| \text{ s.t. } y_k = f_{k|k-1}(w) \;\; k = 1, \ldots, i. \quad (14)$$

Note that, under the NTK regime, we have $f_{k|k-1}(w) \approx f(x_k; w)$, and the solution obtained by ORFit in one pass is the same as that of SGD in the standard multi-pass setting (as characterized in, *e.g.*, [3]). Compared to *NTK-RLS* (13), ORFit in (5) greatly reduces both time and memory complexities from $O(p^2)$ to $O(ip)$. This is particularly important for $p \gg i$, common to the overparameterized settings (for instance, $p \approx 11$M in ResNet-18, commonly used for the CIFAR-10 dataset, consisting of 50K samples).

VII. Conclusion

In this paper, we proposed Orthogonal Recursive Fitting (ORFit) to tackle one-pass learning. We discussed the connection between the proposed method and orthogonal gradient descent (OGD) as well as the recursive least-squares (RLS). Through this connection, we explained the advantages of the proposed method and theoretically characterized its behavior. Our theoretical findings reveal that ORFit attains the same solution as SGD, with much lower time and memory complexities. We validated our method and its theoretical properties through several experiments and discussed its extensions to nonlinear settings, relevant for deep learning.

We conclude with several interesting future directions. First, although ORFit exhibits outstanding performance in the memory-limited setting, some forgetting is still happening. It is caused by the information loss from summarizing the orthogonal basis via IPCA. Theoretically characterizing how much ORFit forgets would be of great importance to understand the practicality of the algorithm. Moreover, one can also come up with other methods to summarize the memory such as matrix sketching. Next, we remark that RLS is a special case of the Kalman filter applied on a static system. Based on this connection, another interesting avenue is to build on ORFit and devise efficient learning/estimation methods for dynamic systems. Lastly, exploring the practicality of ORFit in deep learning based on a more comprehensive set of experiments would be of great interest.

References

[1] Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[2] N. Azizan and B. Hassibi. Stochastic gradient/mirror descent: Minimax optimality and implicit regularization. In *International Conference on Learning Representations*, 2018.

[3] N. Azizan, S. Lale, and B. Hassibi. Stochastic mirror descent on overparameterized nonlinear models. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[4] P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

[5] M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[6] M. Farajtabar, N. Azizan, A. Mott, and A. Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

[7] H. Hu, A. Li, D. Calandriello, and D. Gorur. One pass ImageNet. *arXiv preprint arXiv:2111.01956*, 2021.

[8] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[9] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[10] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

[11] A. Levey and M. Lindenbaum. Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9(8):1371–1374, 2000.

[12] Y. Li and Y. Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in Neural Information Processing Systems*, 31, 2018.

[13] Y. Min, K. Ahn, and N. Azizan. One-pass learning via bridging orthogonal gradient descent and recursive least-squares. *arXiv preprint arXiv:2207.13853*, 2022.

[14] P. Rai, H. Daumé, and S. Venkatasubramanian. Streamed learning: one-pass svms. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1211–1216, 2009.

[15] D. Sahoo, Q. Pham, J. Lu, and S. C. Hoi. Online deep learning: learning deep neural networks on the fly. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2660–2666, 2018.

[16] A. H. Sayed. *Fundamentals of adaptive filtering*. John Wiley & Sons, 2003.

[17] A. Sharma, N. Azizan, and M. Pavone. Sketching curvature for efficient out-of-distribution detection for deep neural networks. In *Uncertainty in Artificial Intelligence*, pages 1958–1967. PMLR, 2021.

[18] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu. Large scale incremental learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 374–382. IEEE, 2019.

[19] T. Yu, C. Zhang, Y. Wang, M. Ma, and Q. Song. Recursive least squares for training and pruning convolutional neural networks. *arXiv preprint arXiv:2201.04813*, 2022.

[20] C. Zhang, Q. Song, H. Zhou, Y. Ou, H. Deng, and L. T. Yang. Revisiting recursive least squares for training deep neural networks. *arXiv preprint arXiv:2109.03220*, 2021.