

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

by

Youngjae Min

B.S. Electrical Engineering and Mathematical Sciences  
Korea Advanced Institute of Science and Technology, 2020

Submitted to the Department of Aeronautics and Astronautics  
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE IN AERONAUTICS & ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2023

© 2023 Massachusetts Institute of Technology. All rights reserved.

Authored by: Youngjae Min

Department of Aeronautics and Astronautics

May 23, 2023

Certified by: Navid Azizan

Esther & Harold E. Edgerton Career Development

Assistant Professor of Mechanical Engineering

Thesis Supervisor

Accepted by: Jonathan P. How

R. C. Maclaurin Professor of Aeronautics and Astronautics

Chair, Graduate Program Committee

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

by

Youngjae Min

Submitted to the Department of Aeronautics and Astronautics  
on May 23, 2023, in partial fulfillment of the  
requirements for the degree of  
Master of Science in Aeronautics and Astronautics

## 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 thesis, 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.

Thesis Supervisor: Navid Azizan

Title: Esther & Harold E. Edgerton Career Development Assistant Professor of Mechanical Engineering

## **Acknowledgments**

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Navid Azizan. His invaluable guidance and unwavering support have been essential throughout my research journey. Reflecting on the past two years of my graduate life, choosing to become his very first student has proven to be one of the best decisions I have ever made.

I would also like to extend my heartfelt appreciation to my friends, who have provided immense support in maintaining my mental well-being and navigating through the relentless stream of deadlines. To those of you who are reading this and were expecting to find your name mentioned, you are absolutely right. I want to express my sincere thanks to each and every one of you.

Lastly, I am deeply grateful to my family, as none of my accomplishments would have been possible without their unwavering support. Mom and Dad, I cannot thank you enough for always being there for me. And to my sister, thank you for being a constant source of motivation in pursuing my dreams.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Contributions . . . . .	8
1.2	Connections to Related Notions . . . . .	9
<b>2</b>	<b>Preliminary</b>	<b>10</b>
2.1	One-Pass Learning . . . . .	10
2.2	Recursive Least-Squares . . . . .	10
2.3	Orthogonal Gradient Descent . . . . .	11
<b>3</b>	<b>Orthogonal Recursive Fitting</b>	<b>13</b>
3.1	Orthogonal Recursive Update . . . . .	13
3.2	Incremental Summary of Memory . . . . .	16
3.3	Vector-Output Model and Batch Update . . . . .	17
3.4	Appendix . . . . .	20
3.4.1	Proof of Lemma 1 . . . . .	20
<b>4</b>	<b>Theoretical Results</b>	<b>22</b>
4.1	Connection to RLS . . . . .	22
4.2	Characterizing the Solution of ORFit . . . . .	23
4.3	Local Optimality to Global Optimality . . . . .	24
4.4	Meaning of Principal Direction . . . . .	25
4.5	Appendix . . . . .	26
4.5.1	Proof of Proposition 2 . . . . .	26

4.5.2	Proof of Theorem 3 . . . . .	27
4.5.3	Proof of Theorem 6 . . . . .	28
4.5.4	Proof of Proposition 8 . . . . .	30
<b>5</b>	<b>Experiments</b>	<b>32</b>
5.1	Learning with Restrictions on Memory Size . . . . .	32
5.2	Learning without Memory Restriction . . . . .	33
<b>6</b>	<b>Extension to Deep Learning</b>	<b>36</b>
6.1	Appendix . . . . .	38
6.1.1	Proof of Theorem 9 . . . . .	38
<b>7</b>	<b>Conclusion</b>	<b>41</b>

# List of Figures

3-1	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 $S$ (which is updated incrementally) 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 $\{(x_k, y_k)\}_{k=1}^{i-1}$ .	14
5-1	Results for the memory-restricted setting (§5.1). (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	34
5-2	Results for the setting without memory restriction (§5.2). (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. The red dashed line indicates the step on which the sample is trained. The shades indicate the standard deviations over 10 independent runs.	35

# Chapter 1

## 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, autonomy, 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, in certain applications, storing the data may be prohibited 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 deep neural networks are prone to entirely forgetting past information while learning new data, which is an issue referred to as “catastrophic forgetting” [14]. 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, and several works have attempted to address them under various contexts. In particular, [12] studied learning the ImageNet dataset in a single pass by revisiting some “important” previous datapoints at each learning step, and [23] 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, [19] 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, [20] 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.*, [21]) tackles one-pass learning for linear models (as elaborated in Section 2.2). 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 such as FNN, CNN, RNN, and LSTM [25, 24]. 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.

## 1.1 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. (§3)

- 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. (§4)
- We demonstrate the practicality of our approach and corroborate our theoretical findings through various experiments. (§5)
- We discuss further extensions to overparameterized nonlinear models, relevant for deep learning. (§6)

## 1.2 Connections to Related Notions

One-pass learning shares some similarities with other settings that deal with streaming data such as *online learning* and *incremental learning*. While these terms are often inconsistently defined in the literature, one may distinguish them based on whether we learn a single datapoint or a batch of data at a time [18]. Although both of these approaches learn from streaming data, they typically assume that data is arriving from the same distribution. Compared to these settings, one-pass learning requires explicit efforts to not alter the predictions on the previous data while learning new data. Thus, it aims to preserve the predictions even when the new data comes from a different distribution. Another related setting is *continual learning* where batches of data from different tasks are sequentially learned [8].

# Chapter 2

## Preliminary

### 2.1 One-Pass Learning

Let  $f(x; w) \in \mathbb{R}^c$  be a model that the agent is trying to fit, where  $x \in \mathcal{X} \subset \mathbb{R}^d$  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  $x_k \in \mathcal{X}$  and  $y_k \in \mathcal{Y} \subset \mathbb{R}^c$ . 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  $f(x; w) = \Phi(x)^\top w = [\phi_1(x) \cdots \phi_c(x)]^\top w$  with feature maps  $\phi_j : \mathcal{X} \rightarrow \mathbb{R}^p$  for  $j \in [c] := \{1, \dots, c\}$ . We then discuss the extension of the results to nonlinear models (*e.g.* overparameterized architectures in deep learning) in Chapter 6.

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

### 2.2 Recursive Least-Squares

First, we briefly review recursive least-squares (RLS) from the control/estimation theory literature (refer to [21] for details). At every step  $i$ , RLS aims to find a parameter

vector that solves the following regularized least-squares problem:

$$w_i^{(RLS)} = \arg \min_w \sum_{k=1}^i (y_k - w^\top x_k)^2 + \|w - w_0\|_{\Pi}^2, \quad (2.1)$$

where  $\|x\|_{\Pi} := \sqrt{x^\top \Pi x}$  for a  $p \times p$  positive-definite matrix  $\Pi$  and  $w_0$  is an initial parameter estimate. Note that the system is underdetermined/overparameterized, and the regularization term in (2.1) is necessary for the solution to be uniquely defined. While there is a closed-form solution for (2.1) 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 of the covariance matrix for every new datapoint. RLS bypasses this issue by computing the new solution  $w_i^{(RLS)}$  of (2.1) 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)* [21]. Consider the following problem:

$$w_i^{(RLS)} = \arg \min_w \sum_{k=1}^i \lambda^{i-k} (y_k - w^\top x_k)^2 + \lambda^i \|w - w_0\|_{\Pi}^2, \quad (2.2)$$

with a forgetting factor  $0 < \lambda \leq 1$ . Note that this reduces to the problem of the vanilla RLS (2.1) when  $\lambda = 1$ . The exact solution of it is recursively updated as follows:

$$\begin{cases} w_i^{(RLS)} = w_{i-1}^{(RLS)} + \frac{P_{i-1} x_i}{\lambda^i + x_i^\top P_{i-1} x_i} (y_i - x_i^\top w_{i-1}^{(RLS)}), \\ P_i = P_{i-1} - \frac{P_{i-1} x_i x_i^\top P_{i-1}}{\lambda^i + x_i^\top P_{i-1} x_i}, \end{cases} \quad (2.3)$$

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  $\Lambda_i = \text{diag}(\lambda^{-1}, \lambda^{-2}, \dots, \lambda^{-i})$ .

## 2.3 Orthogonal Gradient Descent

An algorithm called orthogonal gradient descent (OGD) [9] has been recently proposed in the context of machine learning for a different but related problem. More specif-

ically, [9] considers a *continual learning* setting in which tasks  $\{T_1, T_2, \dots\}$  arrive sequentially, and 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  $f(x; w) \in \mathbb{R}$  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  $\mathcal{G} := \text{span}\{\bigcup_{k=1}^{i-1} \{\nabla_w f(x; w_k)\}_{(x,y) \in T_k}\}$ :

$$\tilde{g} = g - \sum_{v \in S} \text{proj}_v(g), \quad (2.4)$$

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

# Chapter 3

## Orthogonal Recursive Fitting

In this chapter, 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. 3-1 for an illustration.

### 3.1 Orthogonal Recursive Update

We start by considering OGD directly applied to the one-pass learning setting with scalar-output model  $f(x; w) \in \mathbb{R}$ , *i.e.*,  $c = 1$ . 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 [5, 4], it will take many iterations for OGD to perfectly fit the datapoint. Instead, inspired by the recent trend in meta-learning, we 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 Appendix 3.4.1 for a proof.

**Lemma 1.** *Consider a linear model  $f(x; w) = \phi(x)^\top w$ , and let  $\tilde{g}$  be the projection defined by (2.4) of any vector  $g \in \mathbb{R}^p$ . Then, for any step size  $\eta \in \mathbb{R}$ , the new parameter  $w' =$*

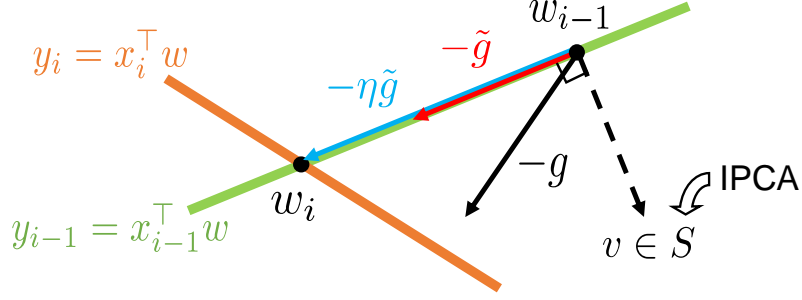


Figure 3-1: An illustration of ORFit in the parameter space for a linear model. The parameter  $w_{i-1}$  fits the previous datapoints  $\{(x_k, y_k)\}_{k=1}^{i-1}$ . The set  $S$  (which is updated incrementally) consists of the directions moving towards which causes the most change in the predictions 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  $\{(x_k, y_k)\}_{k=1}^{i-1}$ .

$w - \eta \tilde{g}$  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  $\eta$  so that the updated parameter  $w'$  can perfectly fit the new datapoint, say  $(x', y')$ , as  $y' = \phi(x')^\top w' = \phi(x')^\top (w - \eta \tilde{g})$ . Following this principle, a straightforward calculation yields the following update rule:

$$\begin{cases} \tilde{g}_{i-1} = \nabla f_i(w_{i-1}) - \sum_{v \in S_{i-1}} \text{proj}_v(\nabla f_i(w_{i-1})), \\ w_i = w_{i-1} - \eta_{i-1} \tilde{g}_{i-1}, \\ S_i = S_{i-1} \cup \{\tilde{g}_{i-1}\}, \end{cases} \quad (3.1)$$

for  $i \geq 1$  where  $S_0$  is the empty set,  $w_0$  is the initial weight vector, and the optimal step size is chosen as

$$\eta_{i-1} = \frac{1}{\nabla f_i(w_{i-1})^\top \tilde{g}_{i-1}} (f_i(w_{i-1}) - y_i). \quad (3.2)$$

Here, we set the direction  $g$  as the model gradient  $\nabla f_i(w_{i-1})$  instead of the loss gradient  $\nabla \ell_i(w_{i-1})$  for simplicity, while they are in parallel to each other. See Algorithm 1 for the

---

**Algorithm 1** Orthogonal Recursive Fitting (ORFit) for scalar-output model ( $c = 1$ ) without memory restriction

---

**Input:** Data sequence  $((x_k, y_k))_{k=1}^K$

**Output:** The optimal parameter  $w$

- 1: **Initialize**  $S \leftarrow \emptyset$ ,  $w \leftarrow w_0$
- 2: **for**  $k = 1, 2, 3, \dots$  **do**
- 3:    $\triangleright$  Update parameter
- 4:    $g \leftarrow$  Sample model gradient  $\nabla f_k(w)$
- 5:    $\tilde{g} \leftarrow g - \sum_{v \in S} \text{proj}_v(g)$
- 6:    $\eta \leftarrow (f_k(w) - y_k) / (g^\top \tilde{g})$
- 7:    $w \leftarrow w - \eta \tilde{g}$
- 8:    $\triangleright$  Update basis
- 9:    $S \leftarrow S \cup \{\tilde{g}\}$
- 10: **end for**

---

detailed procedure. For intuition, we note that the optimal step size (3.2) 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 [2, 17, 1].

*Remark 1* (Computational overhead). It is important to note that all the quantities appearing in (3.2) are typically available in the gradient-based optimization setting, and hence, there is no computational overhead for computing the stepsize (3.2).

*Remark 2* (Nonlinear models). Although the update rule (3.1) 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 Chapter 6.

Another distinction between the update rule (3.1) and OGD lies in the update of the orthogonal basis  $S_i$ : OGD utilizes “fresher” gradient at the updated parameter  $w_i$  by  $S_i = S_{i-1} \cup \{\nabla f_i(w_i) - \sum_{v \in S_{i-1}} \text{proj}_v(\nabla f_i(w_i))\}$ . Although the two bases actually span the same subspace for linear models, it turns out that ORFit in (3.1) leads to a natural generalization for nonlinear models; see Chapter 6 for details.

Although the update rule (3.1) 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.

## 3.2 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 approximate the orthogonal basis  $S$  in a lower dimension using an incremental principal component analysis (IPCA) algorithm, known as the sequential Karhunen–Loeve (SKL) algorithm proposed in [16]. IPCA is a memory-efficient variant of PCA that enables sequential update for streaming/large datasets. Let us formally describe how we utilize IPCA to incrementally approximate the orthogonal basis.

Consider the orthogonal basis  $S$  that spans the past model gradients  $\{g_1, g_2, \dots, g_i\}$  as in (3.1). Let the singular value decomposition (SVD) of  $A = [g_1 \ g_2 \ \dots \ g_i]$  be  $A = U\Sigma V^\top$ . Here, the crucial information of the SVD is the left-singular vectors  $\text{col}(U)$  that forms an orthonormal basis for  $\text{span}(S)$ . This information can be used to come up with a rank- $m$  approximation of  $S$  by using the principal components corresponding to the top  $m$  singular values. This choice of approximation has the particular meaning of minimizing the worst-case forgetting of previous predictions for unknown future updates. We will formally elaborate on the physical meaning of the principal directions in Section 4.4.

Now suppose that the orthogonal basis  $S$  is augmented so that it spans the new gradient  $g'$  as in (3.1). We want to efficiently update  $U$  for the new basis. Letting  $u := g'/\|g'\|$ , the new basis matrix can be represented as:

$$\begin{bmatrix} A & g' \end{bmatrix} = \begin{bmatrix} U & u \end{bmatrix} \begin{bmatrix} \Sigma & U^\top g' \\ 0 & u^\top g' \end{bmatrix} \begin{bmatrix} V^\top & 0 \\ 0 & 1 \end{bmatrix} \quad (3.3)$$

$$= \left( \begin{bmatrix} U & u \end{bmatrix} \tilde{U} \right) \tilde{\Sigma} \left( \tilde{V}^\top \begin{bmatrix} V^\top & 0 \\ 0 & 1 \end{bmatrix} \right), \quad (3.4)$$

where  $\tilde{U}\tilde{\Sigma}\tilde{V}^\top$  is the SVD of  $\begin{bmatrix} \Sigma & U^\top g' \\ 0 & u^\top g' \end{bmatrix}$ . Then, (3.4) 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  $\Sigma$ . The important aspect here is that the update can be made *without storing*  $V$  and

---

**Algorithm 2** Orthogonal Recursive Fitting (ORFit) for scalar-output model ( $c = 1$ ) with memory limit  $m$

---

**Input:** Data sequence  $((x_k, y_k))_{k=1}^K$

**Output:** The optimal parameter  $w$

- 1: **Initialize**  $U \leftarrow [ ]$ ,  $\Sigma \leftarrow [ ]$ ,  $w \leftarrow w_0$
  - 2: **for**  $k = 1, 2, 3, \dots$  **do**
  - 3:    $\triangleright$  Update parameter
  - 4:    $g \leftarrow$  Sample model gradient  $\nabla f_k(w)$
  - 5:    $\tilde{g} \leftarrow g - U(U^\top g)$
  - 6:    $\eta \leftarrow (f_k(w) - y_k)/(g^\top \tilde{g})$
  - 7:    $w \leftarrow w - \eta \tilde{g}$
  - 8:    $\triangleright$  Update basis
  - 9:    $u \leftarrow g/\|g\|$
  - 10:    $\tilde{U}, \tilde{\Sigma} \leftarrow$  Compute SVD of  $\begin{bmatrix} \Sigma & U^\top g \\ 0 & u^\top g \end{bmatrix}$
  - 11:    $U \leftarrow [U \ u] \tilde{U}$
  - 12:    $U, \Sigma \leftarrow$  top  $m$  singular vectors/values in  $U, \Sigma$
  - 13: **end for**
- 

without having to recompute the SVD of the new gradient matrix. Finally, one can store only the top  $m$  singular values in  $\Sigma$  and their corresponding components in  $U$ . By repeatedly applying this IPCA algorithm in addition to (3.1), we obtain *orthogonal recursive fitting* (ORFit). See Algorithm 2 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  $O(mp)$ . Moreover, 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 by appropriately choosing  $m$ .

### 3.3 Vector-Output Model and Batch Update

Now, we extend the algorithm to fit more general vector-output model  $f(x; w) \in \mathbb{R}^c$  which subsumes the scalar-output case ( $c = 1$ ). For concreteness, we consider a linear overparameterized model  $f(x; w) = \Phi(x)^\top w = [\phi_1(x) \cdots \phi_c(x)]^\top w$ . In addition, we further extend the algorithm to learn batches, *i.e.*, chunks, of data. It will turn out that the batch learning can be accomplished in the same manner as we extend to the vector-output model.

*Remark 3* (Other forms of vector-output model). Note that we could also consider another form of vector-output model  $f(x; w_1, \dots, w_c) = [\phi_1(x)^\top w_1 \cdots \phi_c(x)^\top w_c]^\top$  which has a separate parameter vector  $w_j \in \mathbb{R}^p$  for each output dimension  $j \in [c]$ . Especially, with a shared feature vector  $\phi(x) \in \mathbb{R}^p$ , the model could be  $f(x; w_1, \dots, w_c) = [w_1 \cdots w_c]^\top \phi(x)$ . An example is a fully connected neural network that allows tuning only the last layer while freezing the earlier layers. Then, the feature vector  $\phi(x)$  corresponds to the output of the second last layer. However, such model could be separated into multiple scalar-output models, one for each output dimension, with their own parameters. Then, ORFit for  $c = 1$  is enough to update them independently.

We first start by characterizing the update rule (3.1) for the scalar-output model as solving an optimization problem. From its natural interpretation, we generalize the problem to the vector-output model and batch learning setting. Then, we figure out the corresponding generalized update rule. As we will discuss in Chapter 4, for a linear overparameterized model  $f(x; w) = \phi(x)^\top w$ , ORFit in (3.1) finds the same solution as the limiting case of EW-RLS such that

$$w_i = w_{i-1} + \frac{Q_{i-1} \phi(x_i)}{\phi(x_i)^\top Q_{i-1} \phi(x_i)} (y_i - f_i(w_{i-1})), \quad (3.5)$$

where  $Q_{i-1}$  is the projection matrix onto the orthogonal complement of the previous feature space  $\text{span}\{S_{i-1}\} = \text{span}\{\phi(x_k)\}_{k=1}^{i-1}$ . Then, we can rewrite (3.5) as  $\Delta w_i := w_i - w_{i-1} = (\phi(x_i)^\top Q_{i-1})^\dagger (y_i - f_i(w_{i-1}))$  with the Moore–Penrose inverse operation  $(\cdot)^\dagger$ . Also from (3.5),  $\Delta w_i = Q_{i-1} \Delta w_i$  as the projection matrix  $Q_{i-1}$  is idempotent, *i.e.*  $Q_{i-1}^2 = Q_{i-1}$ . Since the Moore–Penrose inverse provides the minimum  $\ell^2$ -norm solutions for under-determined linear systems, (3.5) implies

$$\begin{aligned} \Delta w_i &= \arg \min_{\Delta w} \|\mathcal{Q}_{i-1} \Delta w\|_2 \quad \text{s.t.} \quad y_i - f_i(w_{i-1}) = \phi(x_i)^\top \mathcal{Q}_{i-1} \Delta w \\ &= \arg \min_{\Delta w} \|\mathcal{Q}_{i-1} \Delta w\|_2 \quad \text{s.t.} \quad y_i = \phi(x_i)^\top (w_{i-1} + \mathcal{Q}_{i-1} \Delta w) \\ &= \arg \min_{\Delta w} \|\Delta w\|_2 \quad \text{s.t.} \quad y_i = \phi(x_i)^\top (w_{i-1} + \Delta w) \end{aligned} \quad (3.6)$$

$$\Delta w \perp \text{span}\{S_{i-1}\}$$

The formulation of (3.6) can be naturally generalized to the vector-output model  $f(x; w) = \Phi(x)^\top w$  and data batch sequence  $(\{(x_k^b, y_k^b)\}_{k=1}^{n_b})_{b=1}^B$ . (3.6) finds a new solution that is in the closest  $\ell^2$  distance from the previous one while (i) fitting the new datapoint and (ii) being orthogonal to the previous model gradients. The first constraint would be then generalized to fit the new data batch for the vector-output model:

$$Y_i := \begin{bmatrix} y_1^i \\ \vdots \\ y_{n_i}^i \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi(x_1^i) & \cdots & \Phi(x_{n_i}^i) \end{bmatrix}}_{=: \Phi_i}^\top (w_{i-1} + \Delta w). \quad (3.7)$$

Note that increasing the output dimension  $c$  of the model  $f$  and increasing the batch size  $n_i$  have the same consequence of increasing the number of rows of  $Y_i$  and columns of  $\Phi_i$ . Meanwhile, the role of the second constraint is finding a weight change  $\Delta w$  that is not altering the previous model outputs. It can be generalized by redefining the previous feature space as

$$\text{span}\{\bar{S}_{i-1}\} := \text{span}\{\phi_j(x_k^b) \mid j \in [c], b \in [i-1], k \in [n_b]\}. \quad (3.8)$$

Then, the generalization of (3.6) becomes

$$\begin{aligned} \Delta w_i = \arg \min_{\Delta w} \|\Delta w\|_2 \quad \text{s.t.} \quad & Y_i = \Phi_i^\top (w_{i-1} + \Delta w) \\ & \Delta w \perp \text{span}\{\bar{S}_{i-1}\} \end{aligned} \quad (3.9)$$

The corresponding solution can be analytically written as in (3.5):

$$\Delta w_i = (\Phi_i^\top \bar{Q}_{i-1})^\dagger (Y_i - F_i) = \bar{Q}_{i-1} \Phi_i \left( \Phi_i^\top \bar{Q}_{i-1} \Phi_i \right)^{-1} (Y_i - F_i), \quad (3.10)$$

where  $\bar{Q}_{i-1}$  is the projection matrix onto the orthogonal complement of the subspace  $\text{span}\{\bar{S}_{i-1}\}$  and  $F_i := [f(x_1^i; w_{i-1})^\top \cdots f(x_{n_i}^i; w_{i-1})^\top]^\top$ . This solution can be computed efficiently as described in Algorithm 3.

Similarly in the scalar-output case in Section 3.1, the update rule (3.10) requires storing the orthogonal basis  $\text{col}(U)$ , whose size grows linearly in the number of visited

---

**Algorithm 3** Orthogonal Recursive Fitting (ORFit) for general vector-output model and batch update without memory restriction

---

**Input:** Data batch sequence  $(\{(x_k^b, y_k^b)\}_{k=1}^{n_b})_{b=1}^B$

**Output:** The optimal parameter  $w$

- 1: **Initialize**  $U \leftarrow [ ]$ ,  $w \leftarrow w_0$
  - 2: **for**  $b = 1, 2, 3, \dots$  **do**
  - 3:    $\triangleright$  Update parameter
  - 4:    $J \leftarrow \left[ \left( \frac{df(x_1^b; w)}{dw} \right)^\top \quad \left( \frac{df(x_2^b; w)}{dw} \right)^\top \quad \dots \quad \left( \frac{df(x_{n_b}^b; w)}{dw} \right)^\top \right]$
  - 5:    $\tilde{J} \leftarrow J - U(U^\top J)$
  - 6:    $Y \leftarrow [(y_1^b)^\top (y_2^b)^\top \dots (y_{n_b}^b)^\top]^\top$
  - 7:    $F \leftarrow [f(x_1^b; w)^\top f(x_2^b; w)^\top \dots f(x_{n_b}^b; w)^\top]^\top$
  - 8:    $w \leftarrow w + \tilde{J}(J^\top \tilde{J})^{-1}(Y - F)$
  - 9:    $\triangleright$  Update basis
  - 10:   **for**  $g$  in  $J$  **do**
  - 11:      $\tilde{g} \leftarrow g - U(U^\top g)$
  - 12:      $U \leftarrow [U \ \tilde{g}]$
  - 13:   **end for**
  - 14: **end for**
- 

datapoints. We can resolve this issue by utilizing IPCA as in Section 3.2. See Algorithm 4 for the detailed procedure.

## 3.4 Appendix

### 3.4.1 Proof of Lemma 1

Consider  $(x, y) \in T_k$  for any  $1 \leq k \leq i - 1$ .

$$f(x; w') = \phi(x)^\top w' = \phi(x)^\top (w - \eta \tilde{g}) = f(x; w) - \eta \phi(x)^\top \tilde{g}. \quad (3.11)$$

---

**Algorithm 4** Orthogonal Recursive Fitting (ORFit) for general vector-output model and batch update with memory limit  $m$

---

**Input:** Data batch sequence  $(\{(x_k^b, y_k^b)\}_{k=1}^{n_b})_{b=1}^B$

**Output:** The optimal parameter  $w$

- 1: **Initialize**  $U \leftarrow [ ]$ ,  $w \leftarrow w_0$
  - 2: **for**  $b = 1, 2, 3, \dots$  **do**
  - 3:   ▷ Update parameter
  - 4:    $J \leftarrow \left[ \left( \frac{df(x_1^b; w)}{dw} \right)^\top \quad \left( \frac{df(x_2^b; w)}{dw} \right)^\top \quad \dots \quad \left( \frac{df(x_{n_b}^b; w)}{dw} \right)^\top \right]$
  - 5:    $\tilde{J} \leftarrow J - U(U^\top J)$
  - 6:    $Y \leftarrow [(y_1^b)^\top (y_2^b)^\top \dots (y_{n_b}^b)^\top]^\top$
  - 7:    $F \leftarrow [f(x_1^b; w)^\top f(x_2^b; w)^\top \dots f(x_{n_b}^b; w)^\top]^\top$
  - 8:    $w \leftarrow w + \tilde{J}(J^\top \tilde{J})^{-1}(Y - F)$
  - 9:   ▷ Update basis
  - 10:    $J_{\text{orth}} \leftarrow \text{orthogonalize } \tilde{J}$
  - 11:    $\tilde{U}, \Sigma \leftarrow \text{Compute SVD of } \begin{bmatrix} \Sigma & U^\top J \\ 0 & J_{\text{orth}}^\top \tilde{J} \end{bmatrix}$
  - 12:    $U \leftarrow [U \ J_{\text{orth}}] \tilde{U}$
  - 13:    $U, \Sigma \leftarrow \text{top } m \text{ singular vectors/values in } U, \Sigma$
  - 14: **end for**
- 

Since the orthogonal basis  $S$  spans  $\nabla_w f(x; w_k) = \phi(x)$ ,  $\phi(x)$  can be represented as  $\phi(x) = \sum_{u \in S} \text{proj}_u(\phi(x))$ . Then,

$$\phi(x)^\top \tilde{g} = \left( \sum_{u \in S} \text{proj}_u(\phi(x)) \right)^\top \left( g - \sum_{v \in S} \text{proj}_v(g) \right) \quad (3.12)$$

$$= \left( \sum_{u \in S} \frac{uu^\top}{\|u\|^2} \phi(x) \right)^\top \left( I - \sum_{v \in S} \frac{vv^\top}{\|v\|^2} \right) g \quad (3.13)$$

$$= \phi(x)^\top \left( \sum_{u \in S} \frac{uu^\top}{\|u\|^2} - \sum_{u \in S} \sum_{v \in S} \frac{uu^\top vv^\top}{\|u\|^2 \|v\|^2} \right) g \quad (3.14)$$

$$= \phi(x)^\top \left( \sum_{u \in S} \frac{uu^\top}{\|u\|^2} - \sum_{u \in S} \frac{uu^\top}{\|u\|^2} \right) g = 0 \quad (3.15)$$

as  $u^\top v = 0$  if  $u \neq v$ . Thus,  $f(x; w') = f(x; w)$ . □

# Chapter 4

## Theoretical Results

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

### 4.1 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 Appendix 4.5.1 for a proof.

**Proposition 2.** *Consider a linear overparameterized ( $p \geq K$ ) model  $f(x; w) = x^\top w \in \mathbb{R}$  and a data sequence  $((x_k, y_k))_{k=1}^K$ . 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 EW-RLS update rule (2.3) does with  $\lambda = 0$ ,  $\Pi = I$ , and initialization  $w_0$ . In this setting,  $P_i$  in (2.3) is the projection matrix onto the subspace orthogonal to  $\text{span}\{\nabla f_k(w_{k-1})\}_{k=1}^i$ .*

*Remark 4.* The optimization problem of EW-RLS (2.2) is not well-defined for  $\lambda = 0$ . That said, the update rule (2.3) can be still computed for  $\lambda = 0$ , and ORFit finds the same solution as the limiting case of EW-RLS.

*Remark 5.* ORFit in (3.1) 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 our proposed method, as we discuss next.

## 4.2 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 (2.3) is the solution to the optimization problem (2.2). In light of Proposition 2, one might be tempted to claim that ORFit in (3.1) solves (2.2) with  $\lambda = 0$  and  $\Pi = I$ . However, (2.2) is not well-defined for  $\lambda = 0$ .

Nevertheless, intuitively one can regard ORFit as solving (2.2) in the limit of  $\lambda \rightarrow 0^+$ . Then, for sufficiently small  $\lambda > 0$ , the first term in the objective of (2.2) outweighs the second term, which suggests that, in the overparameterized case, the solution should enforce  $y_k \approx w^\top x_k$  for all  $k = 1, 2, \dots, i$ , while minimizing  $\|w - w_0\|^2$ . In the following theorem, we formalize this intuition and characterize the solution of ORFit; see Appendix 4.5.2 for a proof.

**Theorem 3.** *Consider a linear overparameterized ( $p \geq \sum_{b=1}^B n_b$ ) model  $f(x; w) = \Phi(x)^\top w \in \mathbb{R}^c$  and a data batch sequence  $(\{(x_k^b, y_k^b)\}_{k=1}^{n_b})_{b=1}^B$ . Let  $w_0$  be the initialization and  $m$  be the memory limit. Then, at each iteration  $i$  such that  $\sum_{b=1}^i n_b \leq m$ , the parameter vector obtained by ORFit is the solution of the following optimization problem:*

$$\begin{aligned} w_i = \arg \min_w \quad & \|w - w_0\| \\ \text{s.t.} \quad & y_k^b = f(x_k^b; w) \quad (b \in [i], k \in [n_b]). \end{aligned} \tag{4.1}$$

It is known that, for a linear overparameterized model  $f(x; w) = x^\top w$ , 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 (4.1) with batch size 1 (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  $f(x; w) = x^\top w \in \mathbb{R}$  and a data sequence  $((x_k, y_k))_{k=1}^K$ . 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.*

### 4.3 Local Optimality to Global Optimality

While Theorem 3 characterizes ORFit as solving a global optimization problem, we can also characterize each update step as solving a local optimization problem as in (3.6). Since the orthogonality condition  $\Delta w \perp \text{span}\{S_{i-1}\}$  is satisfied if and only if the previous predictions are preserved. *i.e.*,  $f_k(w_{i-1}) = f_k(w_{i-1} + \Delta w)$  for all  $k \in [i-1]$ , we can recursively characterize each update step as below.

**Proposition 5.** *Consider a linear overparameterized ( $p \geq \sum_{b=1}^B n_b$ ) model  $f(x; w) = \Phi(x)^\top w \in \mathbb{R}^c$  and a data batch sequence  $(\{(x_k^b, y_k^b)\}_{k=1}^{n_b})_{b=1}^B$ . Let  $w_0$  be the initialization and  $m$  be the memory limit. Then, at each iteration  $i$  such that  $\sum_{b=1}^i n_b \leq m$ , the parameter vector obtained by ORFit is the solution of the following optimization problem:*

$$\begin{aligned} w_i = \arg \min_w \quad & \|w - w_{i-1}\|_2 \\ \text{s.t.} \quad & y_k^b = f(x_k^b; w) \quad (b \in [i], k \in [n_b]). \end{aligned} \tag{4.2}$$

Then, the sequence of solving the local optimization problems (4.2) leads to solving the global optimization problem (4.1). This connection between local and global optimality can be understood in a more general setting with the concept of Bregman divergence. For a strictly convex function  $\rho(w)$ , the Bregman divergence is defined as:

$$D_\rho(w, w_{i-1}^*) = \rho(w) - \rho(w_{i-1}^*) - \nabla \rho(w_{i-1}^*) \cdot (w - w_{i-1}^*). \tag{4.3}$$

Then, we can formalize the connection as below. See Appendix 4.5.3 for a proof.

**Theorem 6.** Consider a sequence of local optimization problems for  $i = 1, 2, \dots$

$$\mathcal{P}_i : \quad \begin{aligned} w_i &= \arg \min_w D_\rho(w, w_{i-1}^*) \\ \text{s.t.} \quad & l_i(w) = A_i w - b_i = 0, \end{aligned} \quad (4.4)$$

where  $D_\rho$  is the Bregman divergence for a strictly convex function  $\rho(w)$  and  $w_0$  is the minimum of  $\rho(w)$ . Then, if the later constraints subsume the previous constraints through affine relationships given by  $l_j(w) = E_{j \leftarrow i} l_i(w)$  with a matrix  $E_{j \leftarrow i}$  for all  $j \leq i$ , the optimal solution  $w_i$  of each  $i$ -th local optimization problem  $\mathcal{P}_i$  is also the solution of the following global optimization problem:

$$\arg \min_w \rho(w) \quad \text{s.t.} \quad l_i(w) = 0. \quad (4.5)$$

*Remark 6.* The results for ORFit correspond to a special case with the following setup:

$$\rho(w) = \|w - w_0\|_2, \quad l_i(w) = \Phi_{1:i}^T w - Y_{1:i},$$

where

$$\Phi_{1:i} = \begin{bmatrix} \Phi_1 & \Phi_2 & \dots & \Phi_i \end{bmatrix}, \quad Y_{1:i} = \begin{bmatrix} Y_1^\top & Y_2^\top & \dots & Y_i^\top \end{bmatrix}^\top.$$

## 4.4 Meaning of Principal Direction

ORFit utilizes IPCA to summarize the previous gradients with the top principal components to which the new update is orthogonal. We suggest that the physical meaning of the update direction can be understood in light of the Courant-Fischer-Weyl min-max theorem.

**Lemma 7** (Courant-Fischer-Weyl [10]). *Let  $A = A^T \in \mathbb{R}^{n \times n}$  is a matrix whose eigenvalues are  $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$  and associated orthonormal eigenvectors are  $u_1, u_2, \dots, u_n$ . Also, let  $\mathcal{V}_k$  denotes the set of  $k$ -dimensional subspaces of  $\mathbb{R}^n$ . Then, for each  $1 \leq k \leq n$ ,*

$$\begin{aligned} \lambda_k &= \min_{W \in \mathcal{V}_k} \max_{x \in W, \|x\|=1} x^T A x \\ &= \max_{W \in \mathcal{V}_{n-k+1}} \min_{x \in W, \|x\|=1} x^T A x \end{aligned} \quad (4.6)$$

In addition,  $W = \text{span}\{u_1, \dots, u_k\}$  achieves the outer minimum for the first expression of (4.6), and  $W = \text{span}\{u_1, \dots, u_{k-1}\}^\perp = \text{span}\{u_k, \dots, u_n\}$  achieves the outer maximum for the second expression of (4.6).

*Proof.* This is a standard result of linear algebra related with the Rayleigh quotient and the intersection of subspaces. See, for example, [10, 6, 7, 11] for the complete proof.  $\square$

Utilizing this lemma, we can characterize the meaning of approximating the previous gradients with the top principal directions as minimizing the worst-case forgetting for unknown future updates as below. See Appendix 4.5.4 for a proof.

**Proposition 8.** Consider a linear overparameterized ( $p \geq K$ ) model  $f(x; w) = \phi(x)^\top w \in \mathbb{R}$  and a data sequence  $((x_k, y_k))_{k=1}^K$ . Let  $m$  be the memory limit for ORFit and  $\text{PCA}_m(\cdot)$  denote the span of the top  $m$  left singular vectors. Then, at each iteration  $i \geq m$ , for the accumulated gradients  $J_i := \left[ \nabla f_k(w_{k-1}) \right]_{k=1}^i$ ,  $\text{PCA}_m(J_i)$  is the summary of the memory that minimizes the worst-case forgetting of ORFit in the sense that

$$\text{PCA}_m(J_i) = \arg \min_{S \in \mathcal{V}_m} \max_{\|\Delta w\|=1} \sum_{k=1}^i (f_k(w + \Delta w) - f_k(w))^2, \quad (4.7)$$

s.t.  $\Delta w \perp S$

where  $\mathcal{V}_m$  denotes the set of  $m$ -dimensional subspaces of  $\mathbb{R}^p$ .

## 4.5 Appendix

### 4.5.1 Proof of Proposition 2

In the update rule of ORFit (3.1), the new basis vector can be represented as

$$v' = \nabla f_i(w_{i-1}) - \sum_{v \in \mathcal{S}_{i-1}} \text{proj}_v(\nabla f_i(w_{i-1})) \quad (4.8)$$

$$= \left( I - \sum_{v \in \mathcal{S}_{i-1}} \frac{vv^\top}{\|v\|^2} \right) \nabla f_i(w_{i-1}) = Q_{i-1} x_i, \quad (4.9)$$

where  $Q_{i-1} := I - \sum_{v \in S_{i-1}} vv^\top / \|v\|^2$ . Similarly,

$$\tilde{g}_{i-1} = Q_{i-1} \nabla \ell_i(w_{i-1}) = Q_{i-1} \ell'(y_i, f_i(w_{i-1})) x_i, \quad (4.10)$$

where  $\ell'(\cdot, \cdot)$  denotes the derivative of  $\ell(\cdot, \cdot)$  w.r.t. its second argument.

Putting (4.10) into the parameter update in (3.1),

$$w_i = w_{i-1} + \frac{Q_{i-1} x_i}{x_i^\top Q_{i-1} x_i} (y_i - x_i^\top w_{i-1}). \quad (4.11)$$

Also, note that  $Q_{i-1}$  is symmetric and satisfies  $Q_{i-1}^2 = Q_{i-1}$ , and it corresponds to the projection matrix onto the orthogonal complement of the subspace  $\text{span}\{S_{i-1}\} = \text{span}\{\nabla f_k(w_{k-1})\}_{k=1}^{i-1}$ .

With these properties,  $Q_i$  can be expressed in a recursive form as

$$Q_i = Q_{i-1} - \frac{v' v'^\top}{\|v'\|^2} = Q_{i-1} - \frac{Q_{i-1} x_i x_i^\top Q_{i-1}}{x_i^\top Q_{i-1} x_i}. \quad (4.12)$$

Then, (4.11) and (4.12) are equivalent to the EW-RLS update rule (2.3) with  $\lambda = 0$ ,  $\Pi = I$ , and initialization  $w_0$ , while  $Q_0 = I = P_0$ .  $\square$

## 4.5.2 Proof of Theorem 3

We prove (4.1) using KKT conditions. First, we transform the RHS into an equivalent convex problem:

$$\arg \min_w \frac{1}{2} \|w - w_o\|^2 \text{ s.t. } y_k^b = \Phi(x_k^b)^\top w \quad (b \in [i], k \in [n_b]). \quad (4.13)$$

Let  $Y_{1:i} := [y_1^1 \dots y_{n_i}^i]^\top \in \mathbb{R}^{cn}$  and  $\Phi_{1:i} := [\Phi(x_1^1) \dots \Phi(x_{n_i}^i)] \in \mathbb{R}^{P \times cn}$  where  $n := \sum_{b=1}^i n_b$ .

Then, the Lagrangian of (4.13) is then represented as

$$\mathcal{L}(w, \lambda) = \frac{1}{2} \|w - w_o\|^2 + \lambda^\top (Y_{1:i} - \Phi_{1:i}^\top w). \quad (4.14)$$

Since (4.13) is a convex problem, any primal and dual variables of (4.14) satisfying KKT conditions are primal and dual optimal. The KKT conditions for a pair of primal and dual

variables  $(w^*, \lambda^*)$  are

$$\begin{cases} w^* - w_0 - \Phi_{1:i} \lambda^* = 0 \\ Y_{1:i} - \Phi_{1:i}^\top w^* = 0. \end{cases} \quad (4.15)$$

Now, we show that  $w_i$  from the ORFit update rule (3.10) satisfies (4.15) with some  $\lambda_i$  so that  $w_i$  is the primal optimal solution of (4.13) which shows (4.1). From (3.10),

$$w_i = w_0 + \sum_{b=1}^i \bar{Q}_{b-1} \Phi_b \eta_b, \quad (4.16)$$

where  $\eta_b = (\Phi_b^\top \bar{Q}_{b-1} \Phi_b)^{-1} (Y_b - F_b) \in \mathbb{R}^{c_{n_b}}$ . For each  $b \in [i]$ ,

$$\bar{Q}_{b-1} \Phi_b = \Phi_b - \sum_{v \in \text{col}(\Phi_{1:b-1})} \text{proj}_v \Phi_b. \quad (4.17)$$

Since each projection vector  $v$  is spanned by the columns of  $\Phi_{1:b-1}$ , each column of  $\bar{Q}_{b-1} \Phi_b$  is spanned by the columns of  $\Phi_{1:b}$ . Then, the term  $\bar{Q}_{b-1} \Phi_b \eta_b$  is also spanned by the columns of  $\Phi_{1:b}$ . Thus, the summation term in (4.16) is in total spanned by the columns of  $\Phi_{1:i}$ , *i.e.*, there exists some  $d_i \in \mathbb{R}^{c_n}$  such that

$$w_i = w_0 + \sum_{b=1}^i \bar{Q}_{b-1} \Phi_b \eta_b = w_0 + \Phi_{1:i} d_i. \quad (4.18)$$

By letting  $\lambda_i := d_i$ ,  $(w_i, \lambda_i)$  satisfies the first KKT condition. Also, with the update rule (3.10),  $w_i$  fits all the data  $\{ \{(x_k^b, y_k^b)\}_{k=1}^{n_b} \}_{b=1}^i$ , which implies the second KKT condition. Thus,  $w_i$  is the solution of (4.13) so that (4.1) holds.  $\square$

### 4.5.3 Proof of Theorem 6

By introducing the Lagrange multiplier  $\lambda_i$ , let us define the Lagrangian function associated with  $\mathcal{P}_i$  as

$$\bar{J}_i(w, \lambda) = J_i(w) + \lambda_i^\top l_i(w) = D_\rho(w, w_{i-1}^*) + \lambda_i^\top l_i(w) \quad (4.19)$$

If  $w_i^*$  is an optimal solution of  $\mathcal{P}_i$ , then the first-order necessary condition for optimality implies the existence of multiplier  $\lambda_i^*$  such that

$$\nabla_w \bar{J}_i(w, \lambda_i^*) \Big|_{w_i^*} = \nabla_w D\rho(w, w_{i-1}^*) \Big|_{w_i^*} + \lambda_i^{*\top} \nabla_w l_i(w_i^*) = 0 \quad (4.20)$$

$$\nabla_{\lambda} \bar{J}_i(w_i^*, \lambda) \Big|_{\lambda_i^*} = l_i(w_i^*) = 0 \quad (4.21)$$

To facilitate further analysis, let us recursively define an auxiliary function as

$$L_0(w) = \rho(w) \quad (4.22)$$

$$L_i(w) = L_{i-1}(w) + \lambda_i^{*\top} l_i(w) \quad (i = 1, 2, \dots)$$

Note that  $L_i(w)$  is a strictly convex function in  $w$  since  $\rho(w)$  is strictly convex and  $l_i(w)$  is affine. Since a Bregman divergence is invariant under the affine change in the potential function by definition, we have

$$D\rho(w, w_{i-1}^*) = D_{L_j}(w, w_{i-1}^*) \quad (j = 1, 2, \dots) \quad (4.23)$$

Considering (4.22) and (4.23), (4.20) can be rewritten as

$$\begin{aligned} & \nabla_w \bar{J}_i(w, \lambda_i^*) \Big|_{w_i^*} \\ &= \nabla_w D_{L_{i-1}}(w, w_{i-1}^*) \Big|_{w_i^*} + \nabla_w l_i(w_i^*) - \nabla_w L_{i-1}(w_i^*) \\ &= \nabla_w L_{i-1}(w_i^*) - \nabla_w L_{i-1}(w_{i-1}^*) + \nabla_w l_i(w_i^*) - \nabla_w L_{i-1}(w_i^*) \\ &= \nabla_w l_i(w_i^*) - \nabla_w L_{i-1}(w_{i-1}^*) = 0 \end{aligned} \quad (4.24)$$

A recursive relation arises from (4.24) in that

$$\nabla_w L_i(w_i^*) = \nabla_w L_j(w_j^*) \quad (j = 0, 1, \dots) \quad (4.25)$$

where  $w_j^*$  denotes the solution of  $\mathcal{P}_j$  which satisfies (4.20)-(4.21) for  $i = j$ .

If we define  $w_0^*$  to be the minimizer of  $L_0(w) = \rho(w)$  such that  $\nabla_w L_0(w_0^*) = 0$ , then the recursion relation of (4.25) indicates that  $\nabla_w L_i(w_i^*) = 0$ . The critical point  $w_i^*$  should

be the minimizer of  $L_i(w)$ , because  $L_i(w)$  is a strictly convex function and the optimal solution minimizing a strictly convex objective function is unique. (Note that  $\nabla l_i(w)$  is not a function of  $w$  as  $l_i(w)$  is affine in  $w$ .)

Unrolling the recursion in (4.22) yields

$$L_i(w) = \rho(w) + \sum_{j=1}^i \lambda_j^{*\top} l_j(w) \quad (4.26)$$

Also, if  $l_j(w) = E_{j \leftarrow i} l_i(w)$  holds with some matrix  $E_{j \leftarrow i}$ , then having  $l_i(w_i^*) = 0$  ensures  $l_j(w_j^*) = 0$  for  $j \in [i]$ . By interpreting  $\{\lambda_j^*\}_{j=1}^i$  as a set of Lagrange multipliers associated with multiple affine equality constraints  $\{l_j(w) = 0\}_{j=1}^i$ , we can view  $L_i(w)$  as the Lagrangian function corresponding to the problem in (4.5) whose solution is given by  $w_i^*$ .

Therefore, we have characterized the optimality of  $w_i^*$  with respect to the global problem (4.5), while  $w_i^*$  is the optimal solution for the local subproblem  $\mathcal{P}_i$  at the same time.  $\square$

#### 4.5.4 Proof of Proposition 8

Consider a linear model  $f_i(w) = \phi(x_i)^\top w$ . Suppose that  $N$  datapoints  $x_i$  ( $i \in [N]$ ) are observed, and let  $\Phi := [\phi(x_1) \ \cdots \ \phi(x_N)] \in \mathbb{R}^{p \times N}$ . Let  $\Delta w$  denote the parameter update step. Then, the update direction satisfying  $\Delta w \perp \text{PCA}_m(\Phi)$  minimizes the maximum of the  $\ell_2$ -norm of the function value change spanning all observed datapoints when  $\Delta w$  is confined to an  $(p - m)$ -dimensional subspace of  $\mathbb{R}^p$ .

The change in the function value at a sample  $x_i$  due to parameter update  $\Delta w$  can be written as

$$\Delta f_i(\Delta w) := f_i(w + \Delta w) - f_i(w) = \phi(x_i)^\top \Delta w. \quad (4.27)$$

Concatenating the function value change at all samples into a vector gives

$$\Delta F := \begin{bmatrix} \Delta f_1 \\ \vdots \\ \Delta f_N \end{bmatrix} = \Phi^\top \Delta w. \quad (4.28)$$

Then, we have

$$\sum_{k=1}^i (f_k(w + \Delta w) - f_k(w))^2 = \|\Delta F\|_2^2 = \Delta w^\top \Phi \Phi^\top \Delta w. \quad (4.29)$$

Lemma 7 states that the outer minimum in

$$\begin{aligned} \lambda_{p-m} &= \min_{W \in \mathcal{Y}_{p-m}} \max_{\Delta w \in W, \|\Delta w\|=1} \|\Delta F\|_2^2 \\ &= \min_{W \in \mathcal{Y}_{p-m}} \max_{\Delta w \in W, \|\Delta w\|=1} \Delta w^\top \Phi \Phi^\top \Delta w \end{aligned} \quad (4.30)$$

is achieved with  $W^* = \text{span}(u_1, \dots, u_{p-m})$  where  $u_i$  represents the orthonormal eigenvector of  $\Phi \Phi^\top$  associated with the  $i$ -th least eigenvalue. Since

$$\text{PCA}_m(\Phi) = \text{span}(u_{p-m+1}, \dots, u_p) = W^{*\perp}$$

where  $\perp$  denotes the orthogonal complement of a subspace, an update step  $\Delta w \in W^*$  is orthogonal to the subspace given by  $\text{PCA}_m(\Phi)$ . Therefore,

$$\begin{aligned} \text{PCA}_m(\Phi) &= \arg \min_{S \in \mathcal{Y}_m} \max_{\|\Delta w\|=1} \sum_{k=1}^i (f_k(w + \Delta w) - f_k(w))^2 \\ &\text{s.t. } \Delta w \perp S \end{aligned} \quad (4.31)$$

□

# Chapter 5

## Experiments

In this chapter, we demonstrate the effectiveness of our proposed methods in the one-pass learning setting and corroborate the theoretical results presented in Chapter 4. We performed experiments for linear models in the *Rotated MNIST* setup described in [22]. In this setup, the inputs are rotated MNIST images for digit ‘2’, whose size is  $28 \times 28$ . 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.

### 5.1 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 (3.2) but with empty orthogonal basis. (3) *ORFit-random*: ORFit that keeps 10 randomly chosen basis 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. 5-1a, 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.

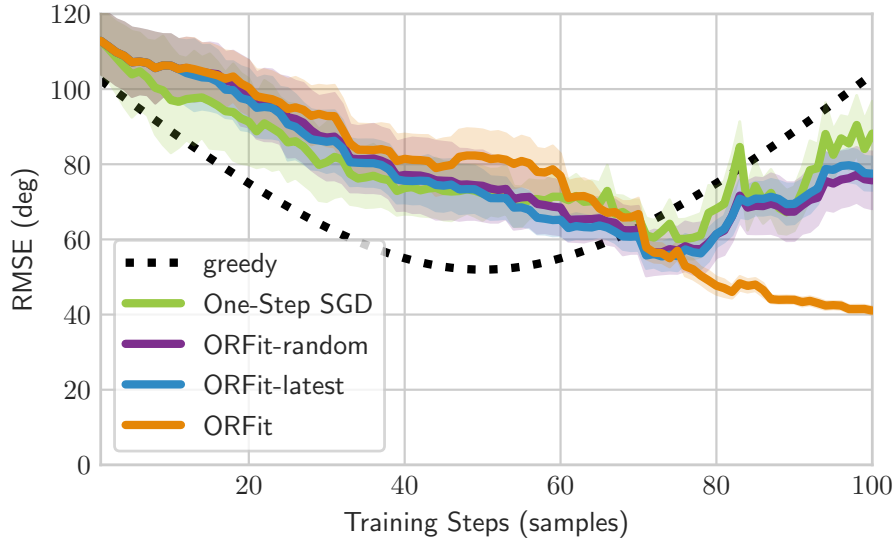
Next, we compared the degrees of forgetting for the proposed method and the baselines by keeping track of the prediction errors of a training datapoint throughout the training. As shown in Fig. 5-1b, ORFit successfully keeps the prediction error low throughout the training. This is in stark contrast with other methods for which the prediction error quickly increases as other datapoints are learned.

## 5.2 Learning without Memory Restriction

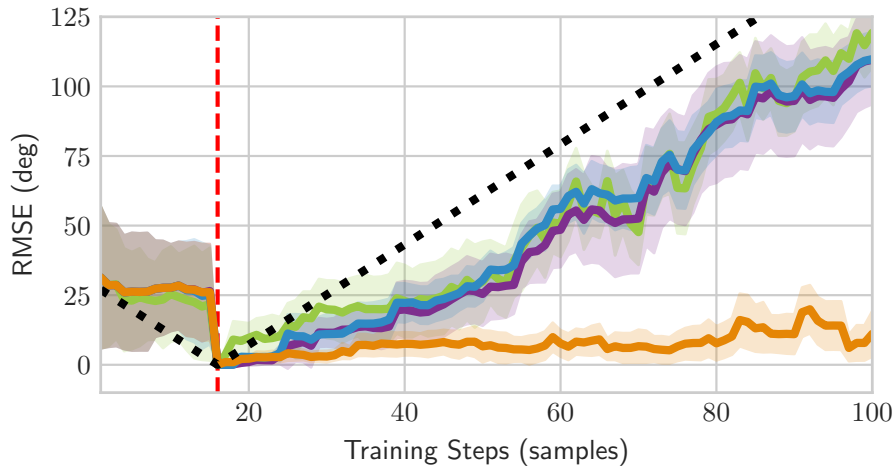
In this experiment, we follow the setup in Section 5.1 except that this time, we do not impose any memory restrictions. For comparison, we run vanilla SGD with a fixed step size  $10^{-5}$ . Vanilla SGD makes multiple passes over the entire dataset for 1000 epochs.

Reported in Fig. 5-2a are the training and test errors of vanilla SGD, One-Step SGD, and ORFit. Note that both SGD and ORFit learn the training 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. 5-2b shows the prediction error of the 11-th training datapoint throughout the training (analogous to Fig. 5-1b). 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.

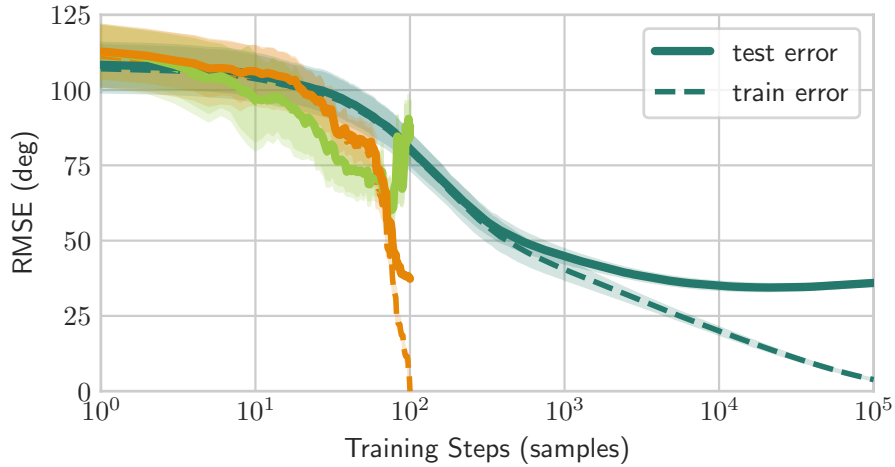


(a) Test Error

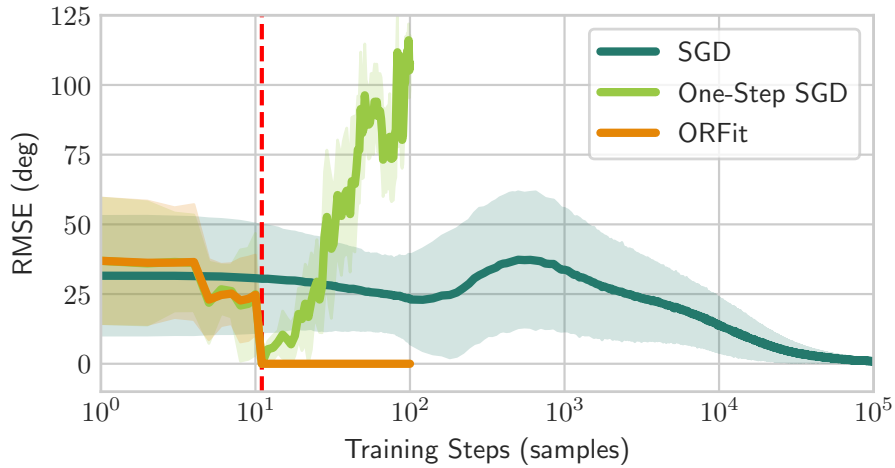


(b) Sample Prediction Error

Figure 5-1: Results for the memory-restricted setting (§5.1). (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



(a) Test & Train Error



(b) Sample Prediction Error

Figure 5-2: Results for the setting without memory restriction (§5.2). (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. The red dashed line indicates the step on which the sample is trained. The shades indicate the standard deviations over 10 independent runs.

# Chapter 6

## Extension to Deep Learning

In this chapter, we extend our discussion to nonlinear models under the neural tangent kernel (NTK) regime [13, 15]. The main idea behind the NTK regime is that when the width of the neural network is chosen large enough, the model is well-approximated by its first-order approximation around the initialization:

$$f_k(w) \approx f_k(w_0) + \nabla f_k(w_0)^\top (w - w_0). \quad (6.1)$$

In particular, Lee et al. [15] 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

$$f_k(w) \approx f_k(w_{k-1}) + \nabla f_k(w_{k-1})^\top (w - w_{k-1}) =: f_{k|k-1}(w). \quad (6.2)$$

Throughout, we denote by  $f_{k|k-1}(w)$  the RHS of (6.2).

A notable feature of ORFit is that it is applicable to any differentiable nonlinear model  $f$ . This is in contrast to the EW-RLS algorithm (2.3), which is only applicable to linear models  $f(x; w) = w^\top x$ . Based on this observation, we employ the step size calculated in (3.2) to fit the new data for a nonlinear model under the NTK regime (6.2). More

specifically, we consider an analog of the EW-RLS (2.2) for nonlinear models:

$$w_i = \arg \min_w \sum_{k=1}^i \lambda^{i-k} (y_k - f_{k|k-1}(w))^2 + \lambda^i \|w - w_0\|_{\Pi}^2, \quad (6.3)$$

given a forgetting factor  $0 < \lambda \leq 1$  and  $\Pi \succ 0$ . Then similarly as in (2.3), one can write out the solution of (6.3) 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:

$$\begin{cases} w_i = w_{i-1} + \frac{P_{i-1} \nabla f_i(w_{i-1}) (y_i - f_i(w_{i-1}))}{\lambda^i + \nabla f_i(w_{i-1})^\top P_{i-1} \nabla f_i(w_{i-1})}, \\ P_i = P_{i-1} - \frac{P_{i-1} \nabla f_i(w_{i-1}) \nabla f_i(w_{i-1})^\top P_{i-1}}{\lambda^i + \nabla f_i(w_{i-1})^\top P_{i-1} \nabla f_i(w_{i-1})}, \end{cases} \quad (6.4)$$

with initialization  $w_0$  and  $P_0 = \Pi^{-1}$ . We call this generalized update rule *NTK-RLS*. Following a similar argument as in Chapter 4, we obtain the following result; see Appendix 6.1.1 for a proof.

**Theorem 9.** *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 (6.4) 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:*

$$\begin{aligned} w_i = \arg \min_w \|w - w_0\| \\ \text{s.t. } y_k = f_{k|k-1}(w) \quad k = 1, 2, \dots, i. \end{aligned} \quad (6.5)$$

Note that, under the NTK regime, for an overparameterized model, 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* (6.4), ORFit in (3.1) 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,  $p \approx 11\text{M}$  in ResNet-18, commonly used for the CIFAR-10 dataset, consisting of 50K samples).

## 6.1 Appendix

### 6.1.1 Proof of Theorem 9

We first prove the connection between ORFit and NTK-RLS. As in the proof of Prop. 2, the new basis vector and the projected gradient of the loss in the update rule of ORFit (3.1) can be represented as

$$v' = Q_{i-1} \nabla f_i(w_{i-1}), \quad (6.6)$$

$$\tilde{g}_{i-1} = Q_{i-1} \ell'(y_i, f_i(w_{i-1})) \nabla f_i(w_{i-1}), \quad (6.7)$$

where  $Q_{i-1} := I - \sum_{v \in \mathcal{S}_{i-1}} v v^\top / \|v\|^2$ . Putting (6.7) into the parameter update in (3.1),

$$w_i = w_{i-1} + \frac{Q_{i-1} \nabla f_i(w_{i-1})(y_i - f_i(w_{i-1}))}{\nabla f_i(w_{i-1})^\top Q_{i-1} \nabla f_i(w_{i-1})}. \quad (6.8)$$

Since  $Q_{i-1}$  is symmetric and satisfies  $Q_{i-1}^2 = Q_{i-1}$ , with (6.6),  $Q_i$  can be expressed in a recursive form as

$$Q_i = Q_{i-1} - \frac{Q_{i-1} \nabla f_i(w_{i-1}) \nabla f_i(w_{i-1})^\top Q_{i-1}}{\nabla f_i(w_{i-1})^\top Q_{i-1} \nabla f_i(w_{i-1})}. \quad (6.9)$$

Then, (6.8) and (6.9) are equivalent to the NTK-RLS update rule (6.4) with  $\lambda = 0$ ,  $\Pi = I$ , and initialization  $w_0$ , while  $Q_0 = I = P_0$ .

We then prove (6.5) using KKT conditions as in Thm. 3. First, we transform the RHS into an equivalent convex problem:

$$\arg \min_w \frac{1}{2} \|w - w_o\|^2 \text{ s.t. } y_k = f_{k|k-1}(w) \quad k = 1, \dots, i. \quad (6.10)$$

Let  $\tilde{y}_k := y_k - f_k(w_{k-1}) + \nabla f_k(w_{k-1})^\top w_{k-1}$ ,  $\tilde{y} := [\tilde{y}_1 \dots \tilde{y}_i]^\top \in \mathbb{R}^i$ , and  $\tilde{X} := [\nabla f_1(w_0) \dots \nabla f_i(w_{i-1})]^\top \in \mathbb{R}^{i \times d}$ . Then, the Lagrangian of (6.10) is then represented as

$$\mathcal{L}(w, \lambda) = \frac{1}{2} \|w - w_o\|^2 + \lambda^\top (\tilde{y} - \tilde{X}w). \quad (6.11)$$

Since (6.10) is a convex problem, any primal and dual variables of (6.11) satisfying KKT

conditions are primal and dual optimal. The KKT conditions for a pair of primal and dual variables  $(w^*, \lambda^*)$  are

$$\begin{cases} w^* - w_0 - \tilde{X}^\top \lambda^* = 0 \\ \tilde{y} - \tilde{X} w^* = 0. \end{cases} \quad (6.12)$$

Now, we show that  $w_i$  from the ORFit update rule (3.1) satisfies (6.12) with some  $\lambda_i$  so that  $w_i$  is the primal optimal solution of (6.10) which in turn shows (6.5). From (3.1),  $w_i = w_0 - \sum_{k=0}^{i-1} \eta_k \tilde{g}_k$ . Then for each  $k$ ,

$$\tilde{g}_k = \nabla \ell_{k+1}(w_k) - \sum_{v \in \mathcal{S}_k} \text{proj}_v(\nabla \ell_{k+1}(w_k)) \quad (6.13)$$

$$= \ell'(y_{k+1}, f_{k+1}(w_k)) \nabla f_{k+1}(w_k) - \sum_{j=1}^k c_{k,j} \nabla f_j(w_{j-1}) \quad (6.14)$$

$$= \tilde{X}^\top d_k, \quad (6.15)$$

where  $d_k := [-c_{k,1}, \dots, -c_{k,k}, \ell'(y_{k+1}, f_{k+1}(w_k)), 0, \dots, 0]^\top \in \mathbb{R}^i$ . Such  $c_{k,j} \in \mathbb{R}$  exists since  $\text{span}\{\mathcal{S}_k\} = \text{span}\{\nabla f_j(w_{j-1})\}_{j=1}^k$ . Then,

$$w_i = w_0 - \sum_{k=0}^{i-1} \eta_k \tilde{X}^\top d_k = w_0 - \tilde{X}^\top \sum_{k=0}^{i-1} \eta_k d_k. \quad (6.16)$$

By letting  $\lambda_i := -\sum_{k=0}^{i-1} \eta_k d_k$ ,  $(w_i, \lambda_i)$  satisfies the first KKT condition.

For the second KKT condition, we observe that for  $k \leq i$ ,

$$f_{k|k-1}(w_i) = f_k(w_{k-1}) + \nabla f_k(w_{k-1})^\top (w_i - w_{k-1}) \quad (6.17)$$

$$= f_k(w_{k-1}) - \nabla f_k(w_{k-1})^\top \sum_{j=k-1}^{i-1} \eta_j \tilde{g}_j \quad (6.18)$$

$$= f_k(w_{k-1}) - \eta_{k-1} \nabla f_k(w_{k-1})^\top \tilde{g}_{k-1} \quad (6.19)$$

$$= f_k(w_{k-1}) - (f_k(w_{k-1}) - y_k) = y_k, \quad (6.20)$$

where (6.19) is satisfied as  $\nabla f_k(w_{k-1}) \perp \tilde{g}_j$  for  $j \geq k$ , and the step-size from (3.2) results in (6.20). Then,  $w_i$  satisfies the second KKT condition and hence is the solution of (6.10).



# Chapter 7

## Conclusion

In this thesis, we proposed an algorithm called Orthogonal Recursive Fitting (ORFit) to tackle one-pass learning. We discussed the connection between the proposed method and orthogonal gradient descent (OGD), a practical algorithm in continual learning, as well as the recursive least-squares (RLS), a well-known method from adaptive filtering. 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 and compare them with ORFit. 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.

# Bibliography

- [1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.
- [2] Navid Azizan and Babak Hassibi. Stochastic gradient/mirror descent: Minimax optimality and implicit regularization. In *International Conference on Learning Representations*, 2018.
- [3] Navid Azizan, Sahin Lale, and Babak Hassibi. Stochastic mirror descent on overparameterized nonlinear models. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
- [4] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.
- [5] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik 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] David Carlson. Minimax and interlacing theorems for matrices. *Linear Algebra and its Applications*, 54:153–172, 1983.
- [7] David Carlson and E. Marquesw De Sa. Generalized minimax and interlacing theorems. *Linear and Multilinear Algebra*, 15(1):77–103, 1984.
- [8] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [9] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.
- [10] Ernst Fischer. Über quadratische formen mit reellen koeffizienten. *Monatshefte für Mathematik und Physik*, 16(1):234–249, 1905.

- [11] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*, chapter 4, pages 234–239. Cambridge University Press, 2 edition, 2012.
- [12] Huiyi Hu, Ang Li, Daniele Calandriello, and Dilan Gorur. One pass ImageNet. *arXiv preprint arXiv:2111.01956*, 2021.
- [13] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.
- [14] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [15] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.
- [16] 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.
- [17] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in Neural Information Processing Systems*, 31, 2018.
- [18] Dinithi Nallaperuma, Rashmika Nawaratne, Tharindu Bandaragoda, Achini Adikari, Su Nguyen, Thimal Kempitiya, Daswin De Silva, Daminda Alahakoon, and Dakshan Pothuhera. Online incremental machine learning platform for big data-driven smart traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4679–4690, 2019.
- [19] Piyush Rai, Hal Daumé, and Suresh Venkatasubramanian. Streamed learning: one-pass svms. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1211–1216, 2009.
- [20] Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH 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.
- [21] Ali H Sayed. *Fundamentals of adaptive filtering*. John Wiley & Sons, 2003.
- [22] Apoorva Sharma, Navid Azizan, and Marco Pavone. Sketching curvature for efficient out-of-distribution detection for deep neural networks. In *Uncertainty in Artificial Intelligence*, pages 1958–1967. PMLR, 2021.

- [23] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 374–382. IEEE, 2019.
- [24] Tianzong Yu, Chunyuan Zhang, Yuan Wang, Meng Ma, and Qi Song. Recursive least squares for training and pruning convolutional neural networks. *arXiv preprint arXiv:2201.04813*, 2022.
- [25] Chunyuan Zhang, Qi Song, Hui Zhou, Yigui Ou, Hongyao Deng, and Laurence Tianruo Yang. Revisiting recursive least squares for training deep neural networks. *arXiv preprint arXiv:2109.03220*, 2021.