

# STOCHASTIC OPTIMIZATION AND LEARNING: AN ADAPTIVE AND RESOURCE-EFFICIENT APPROACH

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Sudeep Salgia

August 2023

© 2023 Sudeep Salgia

ALL RIGHTS RESERVED

STOCHASTIC OPTIMIZATION AND LEARNING: AN ADAPTIVE AND  
RESOURCE-EFFICIENT APPROACH

Sudeep Salgia, Ph.D.

Cornell University 2023

This dissertation focuses on stochastic optimization and learning where the underlying probabilistic models are unknown and the decision maker optimizes their actions over time through sequential interactions with the unknown environment.

The first part of this dissertation is on stochastic optimization where the learner aims at optimizing an unknown random loss function in expectation — the fundamental building block of training any machine learning model today. In the centralized setting, we study three different classes of stochastic optimization problems, categorized based on the structural assumptions of the unknown function. For stochastic convex optimization, we propose the first extension of the coordinate minimization (CM) approach to stochastic optimization. The proposed approach provides a universal framework for extending low-dimensional optimization routines to high-dimensional problems and inherits the scalability and parallelizability properties of CM. For kernel-based optimization, we propose the first algorithm with order-optimal regret guarantees thereby closing the existing gap between upper and lower bounds. Thirdly, for neural net based optimization for contextual bandits, we explore the setting where neural nets are equipped with a smooth activation function, in contrast to the existing work which primarily focuses on ReLU neural nets.

In the second part of this dissertation, we focus on challenges unique to

distributed stochastic optimization. For the problem of distributed linear bandits, we investigate the regret-communication trade-off by establishing the information-theoretic lower bounds on the required communications (in terms of bits) for achieving a sublinear regret order. We also develop an efficient algorithm that is the first to achieve both order-optimal regret and optimal order of communication cost (in bits). We also extend our algorithm for stochastic convex optimization where it continues to enjoy order-optimal regret and communication performance. Secondly, we study the impact of statistical heterogeneity among clients in a distributed kernel-based bandit framework. We adopt a personalization approach to tackle the heterogeneity among users propose an algorithm that achieves order-optimal regret through a novel design that carefully balances personalized exploration with collaborative exploration.

The third part of the dissertation focuses on problems arising in Active Learning and Active Hypothesis Testing. For the problem of online active learning for classifying streaming instances, we develop a disagreement-based learning algorithm for a general hypothesis space and noise model that incurs a bounded regret and has an order-optimal label complexity. We also study the problem of noisy group testing under unknown noise models, which is contrast to the existing studies that assume perfect knowledge of probabilistic model of the noise. We propose a novel algorithm that is agnostic to the noise distribution and offers a sample complexity that adapts to the noise level and is order-optimal in both the population size and the error rate. In the last chapter, we consider the problem of uniformity testing of Lipschitz continuous distributions with bounded support. We propose a sequential test that offers adaptivity to the unknown  $\ell_1$  distance to the uniform distribution, allowing quicker identification of more anomalous (non-uniform) distributions.

## **BIOGRAPHICAL SKETCH**

Sudeep Salgia received his Bachelor of Technology (equivalent to Bachelor of Science) with Honors in Electrical Engineering and Minor in Computer Science from Indian Institute of Technology, Bombay in 2018. He also received the Institute Silver Medal, awarded to the student with best academic performance in the department. He began to pursue his doctorate degree in Electrical and Computer Engineering at Cornell in 2018, right after his Bachelor's degree. He was awarded the Jacobs Scholar Fellowship for the first year at Cornell. During the summer of 2021, he interned at Machine Learning Solutions Lab at Amazon as an Applied Scientist Intern. During his free time, he is interested in outdoor activities like hiking and playing a variety of sports including cricket, badminton and ultimate frisbee.

This dissertation is dedicated to my parents, Shilpa and Deepesh Salgia.

## ACKNOWLEDGEMENTS

This dissertation would not have been possible without the support of numerous people who have been by my side throughout these past five years.

First and foremost, I would like to express by deepest and sincerest gratitude to my advisor, Prof. Qing Zhao. Her incessant support, encouragement and motivation has been my guiding beacon throughout my degree. She has been an amazing mentor and has helped me in all ways I could have asked from an advisor in these past five years. Her patience, warmth and guidance immensely helped me during my difficult spots in Ph.D. and built my character to overcome such difficulties. She also gave me the liberty to explore my research interests which I feel has greatly helped me towards becoming an independent researcher. Her methodical approach and the quest for understanding everything at a fundamental level has left a deep impression on me, motivating me to emulate the same. As her student, I have experienced a very exciting and rewarding Ph.D. I feel fortunate and honoured to have been her student.

Secondly, I would like to thank my collaborators who have worked with me in these past five years. I am very grateful to Sattar Vakili, with whom I have collaborated on several papers, for his mentorship and guidance. I have had several insightful discussions with him, both academic and non-academic, which have helped me improve as a researcher. I am also grateful to him for introducing to me several new directions over the course of my Ph.D., which has helped me broaden my research interests. I would also like to thank Prof. Lang Tong, Prof. Kobi Cohen, Tamir Gabay and Xinyi Wang who have helped me explore interesting ideas beyond the broad scope of my dissertation.

Thirdly, I am very grateful to Prof. Jayadev Acharya, Prof. Vikram Krishnamurthy and Prof. Marten Wegkamp for serving on my committee. I was also

fortunate to take courses with Prof. Acharya and Prof. Wegkamp which introduced me to several new ideas and approaches that have helped me at several stages during my research. The invaluable suggestions and discussions with Prof. Acharya, both inside and outside the classroom have significantly helped me expand my research acumen and horizon.

These past five years would not have been the same with the support of my friends, especially during the two socially distanced years of Covid-19. A very wholehearted thanks to Devesh Khilwani, Himani Sinhmar, Chaitanya Tappu, Swatah Borkotoky, Raunak Sen, and Madhav Mantri for all the amazing time in Ithaca with those fun hikes, dinners and discussions. A special thanks to Parth Kothari, Arka Sadhu, Abhin Shah, Karan Chadha, Kalpesh Krishna and Siddhant Garg for all the laughs, sharing the highs and lows of Ph.D. life and especially for the Zoom game nights during Covid. I would also like to thank all the members of the Cornell Cricket Club, the Ithaca Ultimate Alliance along with Kunal Shastri, James Luo, Sourbh Bhadane, Shanthanu Krishnakumar and Tommy Hentschel for the wonderful games and the lovely memories. I would like to thank Vaidik Shah, Manasi Anand, Karishni Veerabahu, Chaitanya Dasari, Naman Agrawal and all the other people that I have met through the Cornell India Association for the amazing memories. A special thanks to Mayuresh More, Anshaj Khare and Shrav Sibal for all the memes.

I could not have achieved any of this without the continuous love and support of my parents, who have always been on my side, encouraging me to scale greater heights. No amount of words can express my gratitude for them and the role they have played in helping me reach where I stand today. I would also like to express a deep and heartfelt gratitude to Raagini Patki, for her unwavering support and helping me grow as a person and beyond an academic.

## TABLE OF CONTENTS

Biographical Sketch . . . . .	iii
Dedication . . . . .	iv
Acknowledgements . . . . .	v
Table of Contents . . . . .	vii
List of Tables . . . . .	xiv
List of Figures . . . . .	xv
<b>1 Introduction</b>	<b>1</b>
1.1 Stochastic Optimization . . . . .	2
1.1.1 Stochastic Convex Optimization . . . . .	3
1.1.2 Kernel-based Optimization . . . . .	5
1.1.3 Neural Network Optimization . . . . .	6
1.2 Distributed Stochastic Optimization and Learning . . . . .	7
1.3 Active Learning and Hypothesis Testing . . . . .	10
1.4 Outline of the Thesis . . . . .	12
<b>I Stochastic Optimization</b>	<b>14</b>
<b>2 Stochastic Convex Optimization</b>	<b>15</b>
2.1 Introduction . . . . .	15
2.1.1 Stochastic Convex Optimization . . . . .	15
2.1.2 From SGD to SCD . . . . .	16
2.1.3 Main Results . . . . .	19
2.1.4 Related Work . . . . .	20
2.2 Problem Formulation . . . . .	22
2.2.1 The Objective Function . . . . .	22
2.2.2 Regret, Consistency, and Efficiency Measures . . . . .	23
2.3 Progressive Coordinate Minimization . . . . .	24
2.3.1 The General Structure of PCM . . . . .	25
2.3.2 Optimal Design of Constituent Components . . . . .	25
2.4 The Optimal Precision Control . . . . .	28
2.5 Termination Rules . . . . .	30
2.5.1 SGD . . . . .	30
2.5.2 RWT . . . . .	31
2.5.3 The RWT algorithm . . . . .	32
2.5.4 RWT Algorithm for PCM . . . . .	33
2.6 Discussions and Extensions . . . . .	35
2.7 Empirical Results . . . . .	38

<b>3 Kernel-Based Optimization</b>	<b>40</b>
3.1 Introduction . . . . .	40
3.1.1 Gaussian Process Models . . . . .	40
3.1.2 Main results . . . . .	42
3.1.3 Related Work . . . . .	44
3.2 Problem Statement . . . . .	48
3.2.1 Problem Formulation . . . . .	48
3.2.2 Preliminaries on Gaussian processes . . . . .	50
3.3 The GP-ThreDS Algorithm . . . . .	51
3.3.1 Thresholded domain shrinking . . . . .	51
3.3.2 Identifying high-performing nodes . . . . .	54
3.4 Performance Analysis . . . . .	62
3.4.1 Regret Analysis . . . . .	63
3.4.2 Computational Complexity . . . . .	65
3.5 Empirical Studies . . . . .	66
3.5.1 Experiments on Benchmark Functions . . . . .	66
3.5.2 Hyperparameter Tuning of a convolutional neural network	68

<b>4 Neural Network Optimization</b>	<b>73</b>
4.1 Introduction . . . . .	73
4.1.1 Existing Results on Neural Contextual Bandits . . . . .	74
4.1.2 Contribution . . . . .	75
4.1.3 Other Related Work . . . . .	77
4.2 Preliminaries and Problem Formulation . . . . .	78
4.2.1 The Neural Net Model and Corresponding NT Kernel . . . . .	79
4.2.2 Assumptions on Neural Net and NT Kernel . . . . .	80
4.2.3 Information Gain . . . . .	81
4.3 Approximation Error and Confidence Bounds . . . . .	82
4.4 Algorithm . . . . .	85
4.5 Empirical Studies . . . . .	89
4.5.1 Datasets . . . . .	89
4.5.2 Experimental Setting . . . . .	91
4.5.3 Results . . . . .	97

## II Distributed Stochastic Optimization and Learning 101

<b>5 Communication-Efficient Learning for Distributed Stochastic Optimization</b>	<b>102</b>
5.1 Introduction . . . . .	102
5.1.1 Main Results . . . . .	102
5.1.2 Related Work . . . . .	104
5.2 Problem Formulation . . . . .	110
5.3 Progressive Learning and Sharing . . . . .	112

5.3.1	The Basic Structure of PLS . . . . .	112
5.3.2	Detailed Description of PLS . . . . .	114
5.4	Performance Analysis . . . . .	121
5.4.1	Regret Analysis . . . . .	121
5.4.2	Communication Cost . . . . .	124
5.5	Lower Bound on Communication Cost . . . . .	126
5.6	Extensions . . . . .	127
5.6.1	Leveraging Sparsity . . . . .	127
5.6.2	Beyond the Unit Ball . . . . .	131
5.7	From Linear Bandits to Convex Functions . . . . .	133
5.7.1	Basic Structure of CEAL . . . . .	135
5.7.2	The epoch lengths . . . . .	135
5.7.3	The communication strategy . . . . .	137
5.7.4	Performance Analysis . . . . .	138
5.8	Empirical Studies . . . . .	140
5.8.1	PLS . . . . .	141
5.8.2	CEAL . . . . .	144
<b>6</b>	<b>Distributed Learning among Probabilistically Heterogeneous Users</b>	<b>149</b>
6.1	Introduction . . . . .	149
6.1.1	Kernel-based Bandit Problems . . . . .	149
6.1.2	Main Contributions . . . . .	151
6.1.3	Related Work . . . . .	154
6.2	Problem Formulation and Preliminaries . . . . .	155
6.2.1	Problem Formulation . . . . .	156
6.2.2	Preliminaries on GP Models . . . . .	160
6.3	The CEPE Policy . . . . .	162
6.3.1	Algorithm Description . . . . .	162
6.3.2	Performance Analysis . . . . .	165
6.4	Reducing Communication Cost via Sparse Approximation . . . . .	168
6.4.1	Sparse Approximation of GP models . . . . .	169
6.4.2	CEPE with Sparse Approximation . . . . .	169
6.4.3	Performance Analysis . . . . .	171
6.5	Regret Lower Bound . . . . .	173
6.6	Empirical Studies . . . . .	174
6.6.1	Experimental Setup . . . . .	175
6.6.2	Results . . . . .	177
<b>III</b>	<b>Active Learning and Active Hypothesis Testing</b>	<b>179</b>
<b>7</b>	<b>Online Active Learning for Label Efficiency</b>	<b>180</b>
7.1	Introduction . . . . .	180
7.1.1	Previous Work on Active Learning . . . . .	181

7.1.2	Main Results . . . . .	185
7.2	Problem Formulation . . . . .	189
7.2.1	Instances and Hypotheses . . . . .	189
7.2.2	Error Rate, Disagreement, and Bayes Optimizer . . . . .	190
7.2.3	Noise Condition . . . . .	191
7.2.4	Learning Policies and Performance Measure . . . . .	192
7.3	The Online Active Learning Algorithm . . . . .	193
7.3.1	The Basic Structure . . . . .	193
7.3.2	Threshold Design . . . . .	194
7.4	Analysis of Regret and Label Complexity . . . . .	196
7.4.1	Regret . . . . .	197
7.4.2	Label Complexity . . . . .	197
7.4.3	Order Optimality . . . . .	198
7.5	Extensions and Discussions . . . . .	201
7.5.1	Tradeoff Between Label Complexity and Regret . . . . .	201
7.5.2	Implementation for Homogeneous Linear Classification .	208
7.6	Simulation Examples . . . . .	209
<b>8</b>	<b>Adaptive Group Testing Under Unknown Noise Models</b>	<b>214</b>
8.1	Introduction . . . . .	214
8.1.1	Main Results . . . . .	215
8.1.2	Prior Work . . . . .	215
8.2	Problem Formulation . . . . .	217
8.3	Algorithm Description . . . . .	218
8.3.1	The Outer Module . . . . .	219
8.3.2	The Inner Module . . . . .	222
8.3.3	Confidence Based Local Sequential Test . . . . .	224
8.4	Analysis . . . . .	225
<b>9</b>	<b>Sequential Algorithm for Testing Uniformity of Distribution</b>	<b>227</b>
9.1	Introduction . . . . .	227
9.1.1	Main results . . . . .	229
9.1.2	Related Work . . . . .	230
9.2	Uniformity Testing of Discrete Distributions . . . . .	231
9.2.1	Problem Formulation . . . . .	232
9.2.2	Sequential Coincidence Test . . . . .	232
9.2.3	Sample Complexity . . . . .	235
9.3	Uniformity Testing of Continuous Distributions . . . . .	237
9.3.1	Problem Formulation . . . . .	237
9.3.2	Adaptive Binning Coincidence Test . . . . .	238
9.3.3	Sample Complexity . . . . .	240
9.4	Simulations . . . . .	242
<b>Bibliography</b>		<b>245</b>

<b>A Chapter 2</b>	<b>278</b>
A.1 Proof of Theorem 2.4.1 . . . . .	278
A.1.1 Proof of Lemma 2.4.2 . . . . .	282
A.1.2 Proof of Lemma 2.4.3 . . . . .	283
A.2 Proof of Lemma 2.5.1 . . . . .	285
A.2.1 Efficiency of the SGD Routine . . . . .	285
A.2.2 Order Optimality of the Termination Rule . . . . .	287
A.3 Random Walk on a Tree for PCM . . . . .	287
A.3.1 Termination Rule . . . . .	287
A.3.2 Upper Bound on $\mathbb{E}[\tau(\epsilon)]$ . . . . .	290
A.3.3 Regret in One CM Iteration . . . . .	296
A.3.4 Regret Analysis of PCM-RWT . . . . .	297
A.4 Parallel Implementation of PCM . . . . .	298
<b>B Chapter 3</b>	<b>301</b>
B.1 Additional Details on GP-ThreDS Algorithm . . . . .	301
B.1.1 Simplified Implementation of Alg. 4 . . . . .	301
B.1.2 Sequential test with asymmetric confidence levels . . . . .	301
B.2 Detailed Proofs . . . . .	305
B.2.1 Proof of Theorem 3.4.1 . . . . .	306
B.2.2 Proof of Lemma 3.4.2 . . . . .	313
B.2.3 Proof of Lemma 3.4.3 . . . . .	314
B.2.4 Proof of Lemma 3.4.4 . . . . .	317
B.2.5 Proof of Lemma 3.4.7 . . . . .	321
B.2.6 Proof of Lemma B.2.1 . . . . .	322
B.2.7 Proof of Lemma B.2.2 . . . . .	324
<b>C Chapter 4</b>	<b>326</b>
C.1 Proof of Theorem 4.3.1 . . . . .	326
C.1.1 Proof Preliminaries . . . . .	326
C.1.2 Neural Tangent Kernel . . . . .	328
C.1.3 The Activation Function . . . . .	329
C.1.4 Proof of Lemma 4.3.2 . . . . .	331
C.2 Proof of Helper Lemmas . . . . .	340
C.2.1 Proof of Lemma C.1.7 . . . . .	342
C.2.2 Proof of Lemma C.1.8 . . . . .	347
C.2.3 Proof of Lemma C.1.12 . . . . .	348
C.2.4 Proof of Lemma C.1.13 . . . . .	349
C.2.5 Proof of Lemma C.1.14 . . . . .	352
C.2.6 Proof of Lemma C.1.15 . . . . .	354
C.2.7 Proof of Lemma C.2.2 . . . . .	356
C.2.8 Proof of Lemma C.2.3 . . . . .	357
C.2.9 Proof of Lemma C.2.4 . . . . .	358
C.2.10 Proof of Lemma C.2.6 . . . . .	359

C.3	Proof of Theorem 4.3.3 . . . . .	361
C.4	Additional Details on NeuralGCB . . . . .	363
C.4.1	Pseudo Codes . . . . .	364
C.4.2	Proof of Theorem 4.4.1 . . . . .	367
<b>D</b>	<b>Chapter 5</b>	<b>373</b>
D.1	Analysis for PLS . . . . .	373
D.1.1	Proof of Lemma 5.4.2 . . . . .	379
D.1.2	Proof of Lemma 5.4.3 . . . . .	382
D.1.3	Proof of Theorem 5.4.1 . . . . .	383
D.1.4	Proof of Lemma 5.4.5 . . . . .	386
D.1.5	Proof of Theorem 5.4.4 . . . . .	387
D.2	Establishing the Lower Bound . . . . .	388
D.2.1	Proof of Theorem 5.5.1 . . . . .	388
D.3	Sparse PLS . . . . .	398
D.3.1	Proof of Theorem 5.6.1 . . . . .	400
D.4	CEAL . . . . .	401
D.4.1	Proof of Lemma 5.7.2 . . . . .	402
D.4.2	Proof of Lemma 5.7.3 . . . . .	403
<b>E</b>	<b>Chapter 6</b>	<b>406</b>
E.1	Proof of Theorem 6.3.1 . . . . .	406
E.1.1	Proof of Lemma E.1.1 . . . . .	411
E.2	Proof of Theorem 6.4.2 . . . . .	412
E.2.1	Proof of Lemma 6.4.3 . . . . .	415
E.3	Proof of Theorem 6.5.1 . . . . .	418
E.3.1	Preliminaries on constructing functions in RKHS . . . . .	419
E.3.2	Establishing the lower bound on regret . . . . .	420
E.3.3	Extension to the case of $K > 2$ . . . . .	426
E.3.4	Proof of Lemma E.3.1 . . . . .	427
<b>F</b>	<b>Chapter 7</b>	<b>429</b>
F.1	Proof of Theorem 7.4.1 . . . . .	429
F.2	Proof of Theorem 7.4.3 . . . . .	430
F.3	Proof of Theorem 7.4.4 . . . . .	431
F.4	Proof of Theorem 7.4.5 . . . . .	434
F.5	Proof of Theorem 7.5.1 . . . . .	439
F.6	Proof of Theorem 7.5.2 . . . . .	447
<b>G</b>	<b>Chapter 8</b>	<b>451</b>
G.1	Proof of Lemma 8.4.2 . . . . .	451
G.2	Proof of Lemma 8.4.3 . . . . .	454
G.3	Proof of Theorem 8.4.1 . . . . .	458

<b>H Chapter 9</b>	<b>461</b>
H.1 Proof of Theorem 9.2.1 . . . . .	461
H.2 Proof of Theorem 9.3.1 . . . . .	478

## LIST OF TABLES

5.1	Communication cost (in bits) for various algorithms against PLS in their corresponding experimental setup. Reported values are obtained after averaging over 10 Monte Carlo runs. . . . .	143
5.2	Communication cost (in bits) for various algorithms on different datasets. Reported values are obtained after averaging over 10 Monte Carlo runs. . . . .	148

## LIST OF FIGURES

2.1	The sequential test at a sampling point $x$ under sub-Gaussian noise.	33
2.2	Plot of cumulative regret against time for different algorithms for all 9 classification tasks. . . . .	38
3.1	Thresholded domain shrinking. . . . .	51
3.2	An illustration of the random-walk based search. (Node 6 is the single high-performing leaf node. If the random walk is currently at node 2, the correct direction is along the shortest path to node 6: via node 1 and then node 3.) . . . . .	51
3.3	The local sequential test for the decision problem of finding a $\tau$ -exceeding point. . . . .	61
3.4	(a)-(c) Average cumulative regret against wall clock time for different algorithms on benchmark functions ((a)-(b) Rosenbrock, (c) Branin). (d) Computation time (in seconds) for 1000 samples for different algorithms. . . . .	69
3.5	Performance of different algorithms for tuning the hyperparameters of a CNN for image classification. . . . .	71
4.1	Plots with reward function $h_1(x)$ . . . . .	93
4.2	Plots with reward function $h_2(x)$ . . . . .	94
4.3	Plots with reward function $h_3(x)$ . . . . .	96
4.4	Plots for the dataset Mushroom . . . . .	99
4.5	Plots for the dataset Statlog (Shuttle) . . . . .	100
5.1	Cumulative Regret vs Time for different algorithms. The bold line represents the mean obtained over 10 Monte Carlo runs and the shaded region represents the region of error bars corresponding to one standard deviation. . . . .	144
5.2	Cumulative Regret vs Time for different algorithms for on (a) Synthetic dataset and (b) MNIST dataset. The bold line represents the mean obtained over 10 Monte Carlo runs and the shaded region represents the region of error bars corresponding to one standard deviation. The error bars for all algorithms in on MNIST are very small (approximately $\pm 10$ ). . . . .	148
6.1	(a) Cumulative regret of S-CEPE after $T = 2000$ steps relative to CEPE plotted against the reduction in communication offered relative to CEPE when S-CEPE is run with different sizes of inducing sets. (b, c) Cumulative regret incurred by different algorithms over $T = 2000$ steps for different objective functions ((b) Branin, (c) Function in [274]) . . . . .	175
7.1	A typical random walk on a version space tree. . . . .	203

7.2	Comparison with $A^2$ , DHM, and ACAL ( $d = 1$ , Tsybakov noise with $\alpha = 1$ and $c_0 = 5, h^* = h_{0.5}$ ). . . . .	207
7.3	Comparison with $A^2$ , DHM, and ACAL ( $d = 1$ , Tsybakov noise with $\alpha = 0.5$ and $c_0 = 1, h^* = h_{0.5}$ ). . . . .	210
7.4	Comparison with $A^2$ , DHM, and ACAL for ( $d = 2$ , Tsybakov $\alpha = 1$ and $c_0 = 1 h^* = h_{0.25,0.75}$ ). . . . .	211
7.5	Comparison with $A^2$ and DHM for ( $d = 3, N = 50000$ , Tsybakov noise with $\alpha = 1$ and $c_0 = 1, \alpha = 0.5$ and $c_0 = 5, h^* = (1, 0, 0)$ ). . . .	211
7.6	Comparison with $A^2$ and DHM for ( $d = 4, N = 50000$ , Tsybakov noise with $\alpha = 1$ and $c_0 = 1, \alpha = 0.5$ and $c_0 = 5 h^* = (1, 0, 0, 0)$ ). . .	212
7.7	Comparison with CB-C-G [61] under Tsybakov noise with $\alpha = 0.5$ and $c_0 = 1$ . . . . .	212
8.1	This is a representative figure for the outer module on a population of $n = 8$ items that contain $d = 2$ defective items (shown in red). The random walk begins at the root node. In the first step, it errs and zooms into the left child with 4 items (shown by a dotted arrow). Realizing the current node is not a parent of $\mathcal{D}$ , it zooms back to its parent (here, $\mathcal{S}$ ). In the next step, it correctly zooms into the right child followed by zooming into the target node in the step. It confirms that it is indeed the target node using a leaf level local sequential test. . . . .	220
8.2	The above figure explains the working of the inner module on a set with $m = 4$ items and $d = 2$ defectives (shown in red). The left and right trees represent the first and the second iterations of the random walk respectively. In the first iteration, the random walk zooms into the correct child, makes a mistake, rectifies the same and identifies a defective item correctly. In the second iteration, it does not consider the identified item and carries out another random walk to identify the other defective item. Once both the defectives are identified, they are combined to output the final set $\mathcal{A}'$ . . . . .	221
9.1	True Positive Rate (right) and Empirical Sample Complexity (left) vs $\ell_1$ distance for discrete alternative distributions. . . . .	241
9.2	True Positive Rate (right) and Empirical Sample Complexity (left) vs $\ell_1$ distance for continuous alternative distributions. . . .	243
9.3	True Positive Rate (right) and Empirical Sample Complexity (left) vs $\ell_1$ distance for alternative samples collected from simulated power system. . . . .	244

## CHAPTER 1

### INTRODUCTION

Sequential Learning refers to the class of learning problems where the data are made available to the learner on the fly, through an online mechanism. In typical sequential learning tasks, the learner aims to learn about some property of an unknown environment, through repeated interactions with the environment. As learning proceeds, every interaction with the environment provides a new datum to the learner thereby sequentially augmenting the data available at the learner's disposal. A classical example of a sequential learning task is the estimation of the bias of a coin, i.e., the probability of seeing a heads when the coin is tossed. In this example, the unknown environment is the coin and the quantity of interest is its bias. The learner can interact with the coin (environment) by repeatedly tossing it. Every time the learner tosses the coin, it obtains a new sample (datum), which it can then use to estimate the bias of the coin.

Sequential learning goes far beyond this simple task of estimating the bias of a coin. It has been extensively studied and used over the years in various fields including stochastic optimization, reinforcement learning and game theory. It continues to be an active area of research with numerous applications in the fields of advertisement, finance, automation and healthcare among others. In this thesis, we focus on two important and very widely studied classes of sequential learning tasks — Stochastic Optimization, in both centralized and distributed settings, and Hypothesis Testing.

## 1.1 Stochastic Optimization

Stochastic Optimization aims at optimizing, i.e., minimizing or maximizing, a random loss function  $F(\mathbf{x}; \xi)$  in expectation:

$$f(\mathbf{x}) = \mathbb{E}_\xi[F(\mathbf{x}; \xi)],$$

where  $\mathbf{x}$  is the decision variable that belongs to the domain  $\mathcal{X} \subset \mathbb{R}^d$ , usually assumed to be convex and compact.  $\xi$  is an endogenous random vector whose probabilistic model is unknown, or even when it is known, the expectation of  $F(\mathbf{x}; \xi)$  over  $\xi$  cannot be analytically computed. As a result, the objective function  $f(\mathbf{x})$  is unknown. With the objective function known, the learner can only take a trial-and-error learning approach by choosing, sequentially in time, a sequence of query points  $\{\mathbf{x}_t\}_{t=1}^\infty$  with the hope that the decisions improve over time and convergence towards an optimizer of  $f$ . Various error feedback models have been considered. The zeroth-order vs. first-order feedback pertains to whether the random loss or its (random) gradient at each query point is used in the learning algorithm. The full-information vs. bandit feedback relates to whether the entire loss function over all  $\mathcal{X}$  or only the random loss/gradient at the queried point is revealed at each time.

The archetypal statistical learning problem of classification based on random instances is a stochastic optimization problem, where the decision variable  $\mathbf{x}$  is the classifier and  $\xi$  corresponds to the chosen random instance consisting of a feature vector and its corresponding hidden label. In fact, training any machine learning model by minimizing its loss function based on data samples is an instance of stochastic optimization.

Without any structural assumptions on the unknown function, any hope to

solve the problem is completely forgone. Thus, to ensure feasibility of the aforementioned optimization problem, one assumes certain regularity conditions on the unknown objective function. Based on the structure endowed with the unknown function, the existing literature can be broadly classified into three categories. The first of such studies the problem where that underlying function is convex. The second school of thought is concerned with optimization of functions belonging to a Reproducing Kernel Hilbert Space (RKHS), which is a family of functions satisfying certain regularity conditions imposed by the associated kernel. The third class considers the optimization of functions that can be modelled using a neural network. We provide below a brief description of the current state of literature, the motivating questions and our contributions in each of these three classes of optimization problems.

### 1.1.1 Stochastic Convex Optimization

Stochastic convex optimization was pioneered by Robbins and Munro in 1951 [239], who studied the problem of approximating the root of a monotone function  $g(\mathbf{x})$  based on successive observations of noisy function values at chosen query points. The problem was originally referred to as stochastic approximation, later also known as stochastic root finding. Its equivalence to stochastic convex optimization is immediate when  $g(\mathbf{x})$  is viewed as the gradient of a convex function  $f(\mathbf{x})$  to be minimized. The Stochastic Gradient Descent (SGD) approach developed by Robbins and Munro has long become a classic and continues to be widely used till today. The basic idea of SGD is to choose the next query point  $\mathbf{x}_{t+1}$  in the opposite direction of the observed gradient at the current query point  $\mathbf{x}_t$ . Numerous variants of SGD have since been developed and their

performance has been analyzed under various measures.

Despite the popularity of SGD and variants, a major drawback of SGD is the requirement of manual control of the learning rate. The optimal choice of the learning rate depends on strong convexity and smoothness parameters of the objective function, both of which are usually unavailable in practice. Moreover, the high-cost in computing full gradients in large-scale high-dimensional problems and the resistance of SGD to parallel and distributed implementation have prompted the search for alternative approaches that enjoy parallelizability and scalability.

In this thesis, we propose a two-step solution to address these shortcomings. In the first step, we propose a new stochastic optimization algorithm for one-dimensional settings that is robust, has no tuning parameters and self-adapts to the unknown function characteristics while achieving the optimal learning efficiency. The second step of solution resolves the issue of parallelizability through a novel framework called Progressive Coordinate Minimization (PCM) [253]. PCM is rooted in the methodology of decomposing a high-dimensional optimization problem into a sequence of low-dimensional problems which can be implemented in parallel. PCM, when used in conjunction with an arbitrary low-dimensional optimization routine, for example like the one proposed earlier or even SGD, provides a general, routine-agnostic framework for a scalable, parallelizable and computationally-efficient high-dimensional extension of the low-dimensional routine, that retains the desirable characteristics of the original routine. The extension of SGD within the PCM framework leads to a marriage between the efficiency of SGD with the scalability and parallelizability offered by PCM.

### 1.1.2 Kernel-based Optimization

In kernel-based optimization, the unknown objective function  $f$  is assumed to live in a Reproducing Kernel Hilbert Space (RKHS) of a known kernel. At a high level, RKHS is a Hilbert space of functions, where each function can be evaluated at any point by taking an inner product with a function determined by the kernel and point of interest. The advantage of RKHS based modelling lies in the expressive power of this family of functions, which makes them a useful tool to model black-box functions, especially those that are expensive to evaluate can be accessed only through their values at the queried points. Consequently, kernel-based optimization is often studied in the context of maximizing a black-box function with application in tuning hyperparameters in experiment design, optimizing control strategies in complex systems, neural networks, and scientific simulation based studies.

The high flexibility and expressivity of RKHS, while beneficial for modelling the unknown objective function, makes kernel-based optimization significantly more challenging than stochastic convex optimization. The literature, prior to our work, largely consisted of an array of heuristics with little to no theoretical guarantees. In 2010, Srinivas et al., proposed a new algorithm inspired by the popular Upper Confidence Bound strategy in bandit literature for kernel-based optimization. Their algorithm, despite being suboptimal, brought forth new ideas and pioneered a flurry of new research in this field. However, an optimal algorithm continued to remain elusive. Furthermore, a common drawback among all approaches was their intensive computational complexity rendering them practically infeasible even for problems with moderate dimension.

In Chapter 3, we systematically address these gaps in our fundamental un-

derstanding of this challenging setup of kernel-based optimization. For the noisy setup, we propose a novel algorithm [247] that achieves optimal performance guarantees in kernel-based optimization, thereby closing the gap between the best known lower and upper bounds on performance for kernel-based optimization algorithms. At the same time, the proposed algorithm also offers significant reduction in computational costs, thereby improving upon the state of the art simultaneously on both the theoretical and as well as computational fronts.

### 1.1.3 Neural Network Optimization

In recent years, there has been an increasing effort to understand the optimization of functions modelled using a neural net to better understand and explain the unparalleled performance of neural nets across a variety of tasks, especially in the overparametrized regime. A celebrated work by Jacot *et al.* [147] shows that under a Gaussian initialization of the neural net weights, the output of the neural net converges to a Gaussian Process corresponding to a kernel determined by the architecture and activation function, as the width of the network becomes infinitely large. This correspondence had built a bridge between optimization of neural nets and kernel-based optimization, allowing one to adopt techniques in the latter to that in the former.

A crucial tool that allows leveraging techniques from kernel-based optimization to optimization of neural nets is the error bound between the approximate kernel corresponding to a *finite* neural net and the kernel corresponding to the infinite width limit. So far, these bounds have only been derived

for neural nets with ReLU activation function, which results in a kernel with limited smoothness. Extending these results to smoother activation functions, which are known to be helpful to model smoother functions and also converge to smoother kernels, would allow one to obtain better convergence rates for optimization of neural nets. Moreover, existing algorithms that are designed on this kernel-based viewpoint of neural nets known to have either sub-optimal theoretical guarantees or poor practical performance, resulting in a gap between theory and practice.

Our work [249] has been focused on resolving these two aforementioned issues. Specifically, we derive non-asymptotic error bounds between approximate kernel corresponding to a finite width neural net with general smooth activation functions and corresponding kernel obtained in the infinite width limit. This allows one to adopt kernel based approaches for neural nets with activation functions with arbitrary smoothness, resulting in significantly faster convergence to the optimizer. We also propose a new algorithm that works with all levels of smoothness of activation functions, achieves optimal performance bounds in theory and also outperforms existing algorithms in several empirical studies.

## 1.2 Distributed Stochastic Optimization and Learning

Recent times have witnessed an exponential rise in data generation and collection, which has necessitated the storage of data across several devices for easier retrieval and management. In addition, emergence of newer applications like Federated Learning where the data needs to be physically separated for privacy

and security reasons has resulted in decentralization of data associated with a learning task. As data gets more decentralized, learning algorithms need to incorporate a collaborative effort from various devices to achieve the learning objective. Distributed Stochastic Optimization refers to this strategy of employing several networked entities, called clients or agents, to collaboratively optimize an unknown objective function.

Distributed Stochastic Optimization brings forth a new set of challenges that are not encountered in the traditional, centralized settings considered in the previous section. Arguably the major challenge in design of distributed optimization algorithms is the control of communication overhead. Since data is decentralized and multiple devices contributed towards the optimization process, communication is essential for collaboration and exchanging information among the clients. Clearly, better accuracy is achieved with more information exchange which contributes towards a greater communication overhead. This results in an accuracy-communication trade-off. While there has been a considerable effort towards designing communication-efficient algorithms, this accuracy-communication trade-off is yet not understood at a fundamental, information-theoretic level. Another challenge, which is more typical in Federated Learning (FL), is the heterogeneity of data distribution across devices. In FL, the participants are typically single-user devices, e.g. cellphones, and hence have private data corresponding to a user-dependent distribution. While the availability of more data benefits the optimization process at a macro-level, the challenge arises at the micro-level where a single global optimizer may not be able to simultaneously minimize the objective functions corresponding to a variety of user-dependent distributions.

In Chapter 5, we undertake a systematic study to rigorously analyze the accuracy-communication trade-off frontier at an information-theoretic level for the representative problem of linear bandits, which is optimization of a special case of convex functions. Our results establish the minimum number of bits that need to be communicated to achieve the optimal performance. We also propose a new algorithm [251] based on progressive learning and sharing that achieves order-optimal performance in terms of both accuracy and communication cost, as dictated by our lower bounds. Building upon this work, we extend the ideas to distributed optimization of strongly convex functions where our proposed algorithm that again achieves optimal regret performance and near-optimal communication cost, significantly outperforming the popular, existing approaches for distributed optimization.

We also work towards addressing the second challenge of data heterogeneity among clients in the kernel-based optimization setup. We adopt a personalization framework to combat the effect of data heterogeneity. Within this framework, we propose a novel algorithm [248] that strikes a careful balance between data sharing through altruistic actions taken by clients (collaborative exploration) and boosting local performance at each client (personalized exploitation). The proposed algorithm achieves a low communication cost while simultaneously achieving order-optimal performance, as corroborated by our lower bounds.

### 1.3 Active Learning and Hypothesis Testing

Hypothesis Testing refers to the class of problems where the objective of the learner is to identify the true hypothesis, from a set of two or more hypotheses, that corresponds to the underlying data. Sequential Hypothesis Testing, also referred to as the optimal design of experiments or active learning, is a sequential version of this problem where the decision maker can *actively* choose which data points to use or which experiments to carry out in order to maximize the relevant information to identify the true hypothesis while minimizing the number of data points or experiments. Such strategies that *actively* seek out relevant, informative data points to quickly achieve the learning objective are particularly important in this era of Big Data when the over abundance of data may well become an obstacle to gathering meaningful information.

For the problem of classification, which is a hypothesis testing problem over a set of classifiers, there has been a considerable effort for developing theories in the offline setting, where all the data is available prior to the deployment of the learning algorithm. However, in practice, the problem is inherent online or sequential — as more data gets collected, more informative points become available, requiring algorithms to dynamically update their query strategy for future observations. Despite its pervasiveness, this online/sequential setting has received lesser attention than its offline counterpart.

We study the problem of learning the best classifier from a given hypothesis class by *actively* querying from a sequence of incoming data points. The objective in such settings is to design algorithms that simultaneously minimize the number of queried data points along with the cumulative excess prediction

error over the best classifier. The sequential setting along with the control of cumulative error as opposed to that of final classifier makes this problem significantly more challenging than its offline counterpart. In this challenging setup, we propose a new algorithm [141] that makes no more than a fixed number of mistakes over the best classifier, irrespective of the number of observed data points by querying the smallest feasible order of data points.

In addition to the task of classification in statistical learning, sequential hypothesis testing finds applications several other areas like group testing and anomaly detection. An innate advantage of sequential approach in such tasks is its ability to offer adaptivity to complexity of the problem instance i.e., allowing simpler tasks to be solved faster, without the prior knowledge about the complexity. Such adaptivity is particularly beneficial for deployment as the algorithm can continuously adapt to new tasks without being required to be tuned for each new task. Despite the practical advantages, designing adaptive algorithms for group testing and anomaly detection continues to remain an unexplored territory.

In the last two chapters of this thesis, we explore designing adaptive algorithms for group testing and anomaly detection. In Chapter 8, we study the problem of group testing, where the objective is quickly and reliably identify the anomalous subset from a given set of items by judiciously choosing which subsets to test at each time. Our work is aimed at remedying the unrealistic assumption of knowing the probabilistic model of noise that is widely adopted in existing algorithms. To this effect, we propose a new group testing algorithm [250] that works for general observation models, is agnostic to the noise distribution, *adapts* to the signal to noise ratio and achieves order-optimal de-

tection delay with respect to the noise level and detection accuracy.

In Chapter 9, we focus on the problem of anomaly detection in the context of distinguishing whether the observed data corresponds to a known, nominal distribution or to a distribution at least  $\epsilon$  far away from the nominal distribution. For this problem of distribution testing, we propose a new sequential algorithm [252] whose sample complexity adapts to the underlying problem instance. In particular, the algorithm requires lesser time (or equivalently, samples) to detect as the data-generating distribution gets further away from the nominal one. Such adaptivity is particularly useful when the class of alternative hypotheses represent anomalies in critical infrastructure where it is often crucial to detect quickly those severe anomalies far away from the normal state, as more severe anomalies often carry more risk.

## 1.4 Outline of the Thesis

The thesis is divided into three parts — Stochastic Optimization, Centralized and Distributed, and Hypothesis Testing, with each part further divided into chapters, resulting in a total of 8 chapters. Each chapter is associated with addressing a particular question of interest and consists of four sections broadly categorized along the following lines. The first section motivates the problem and provides a comparison with existing studies in similar areas. The second one provides a technical description of the problem. The next two sections provide the technical description of the main contribution and the associated theoretical guarantees. Certain chapters also contain an additional fifth section documenting the associated empirical study. The proofs of all theorems and

lemmas are documented in the Appendix at the end of the thesis.

Throughout the thesis, the set of natural numbers, integers and real numbers are denoted by  $\mathbb{N}$ ,  $\mathbb{Z}$  and  $\mathbb{R}$  respectively.  $\mathcal{X} \subset \mathbb{R}^d$  denotes the domain of the function of interest, where  $d$  is used to denote the dimension of the underlying space. Unless specified otherwise,  $\mathcal{X}$  is assumed to be a convex, compact set. Other calligraphic letters are used to denote sets. Given any set  $\mathcal{A}$ ,  $\mathbb{1}(\mathcal{A})$  corresponds to the indicator of the set  $\mathcal{A}$ . We say that a function  $f(x) = O(g(x))$  if there exist a constant  $M > 0$  and  $x_0 \in \mathbb{R}$  such that  $|f(x)| \leq M|g(x)|$  for all  $x \geq x_0$ . Moreover, we use  $f(x) = \tilde{O}(g(x))$  to refer to the case where  $|f(x)| \leq M|g(x)| \cdot |h(\log x)|$ , where  $h(x)$  is a polynomial function. Similarly, we say that  $f(x) = \Omega(g(x))$  if there exist a constant  $m > 0$  and  $x_0 \in \mathbb{R}$  such that  $|f(x)| \geq m|g(x)|$  for all  $x \geq x_0$ . If there exists a  $g(x)$  such that  $f(x) = O(g(x))$  and  $f(x) = \Omega(g(x))$ , then we say that  $f(x)$  satisfies  $f(x) = \Theta(g(x))$ .

## **Part I**

# **Stochastic Optimization**

## CHAPTER 2

### STOCHASTIC CONVEX OPTIMIZATION

## 2.1 Introduction

### 2.1.1 Stochastic Convex Optimization

Stochastic convex optimization aims at minimizing a random loss function  $F(\mathbf{x}; \xi)$  in expectation:

$$f(\mathbf{x}) = \mathbb{E}_\xi [F(\mathbf{x}; \xi)], \quad (2.1)$$

where  $\mathbf{x}$  is the decision variable in a convex and compact set  $\mathcal{X} \subset \mathbb{R}^d$  and  $\xi$  is an endogenous random vector. The probabilistic model of  $\xi$  is unknown, or even when it is known, the expectation of  $F(\mathbf{x}; \xi)$  over  $\xi$  cannot be analytically characterized. As a result, the objective function  $f(\mathbf{x})$  is unknown.

With the objective function unknown, the decision maker can only take a trial-and-error learning approach by choosing, sequentially in time, a sequence of query points  $\{\mathbf{x}_t\}_{t=1}^T$  with the hope that the decisions improve over time. Various error feedback models have been considered. The zeroth-order vs. first-order feedback pertains to whether the random loss  $F(\mathbf{x}_t; \xi_t)$  or its gradient  $G(\mathbf{x}_t; \xi_t)$  at each query point  $\mathbf{x}_t$  is used in the learning algorithm. The full-information vs. bandit feedback relates to whether the entire loss function  $F(\mathbf{x}; \xi_t)$  over all  $\mathbf{x}$  or only the random loss/ gradient at the queried point  $\mathbf{x}_t$  is revealed at each time.

The performance measure has traditionally focused on the convergence of  $\mathbf{x}_T$  to the minimizer  $\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$  or  $f(\mathbf{x}_T)$  to  $f(\mathbf{x}^*)$ . In an online setting, a more

suitable performance measure is the cumulative regret defined as the expected cumulative loss at the query points in excess to the minimum loss:

$$R(T) = \mathbb{E} \left[ \sum_{t=1}^T (F(\mathbf{x}_t; \xi_t) - f(\mathbf{x}^*)) \right]. \quad (2.2)$$

This performance measure gives rise to the exploration-exploitation tradeoff: the need to explore the entire domain  $\mathcal{X}$  for the sake of future decisions and the desire to exploit the currently best decision indicated by past observations to reduce present loss. A learning algorithm with a sublinear regret order in  $T$  implies the convergence of  $f(\mathbf{x}_T)$  to  $f(\mathbf{x}^*)$ , and the specific order measures the rate of convergence.

The archetypal statistical learning problem of classification based on random instances is a stochastic optimization problem, where the decision variable  $\mathbf{x}$  is the classifier and  $\xi$  the random instance consisting of its feature vector and hidden label. The probabilistic dependence between feature and label is unknown. Another example is the design of large-scale complex physical systems that defy analytical modeling. Noisy observations via stochastic simulation is all that is available for decision making.

### 2.1.2 From SGD to SCD

Stochastic convex optimization was pioneered by Robbins and Monro in 1951 [239], who studied the problem of approximating the root of a monotone function  $g(\mathbf{x})$  based on successive observations of noisy function values at chosen query points. The problem was originally referred to as stochastic approximation, later also known as stochastic root finding [221]. Its equivalence to the first-order stochastic convex optimization is immediate when  $g(\mathbf{x})$  is viewed as

the gradient of a convex function  $f(\mathbf{x})$  to be minimized. The zeroth-order version of the problem was studied by Keifer and Wolowitz [165]. The stochastic gradient descent (SGD) approach developed by Robbins and Monro [239] has long become a classic and is widely used. The basic idea of SGD is to choose the next query point  $\mathbf{x}_{t+1}$  in the opposite direction of the observed gradient while ensuring  $\mathbf{x}_{t+1} \in \mathcal{X}$  via a projection operation. Omitting the projection operation, we can write,

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t G(\mathbf{x}_t, \xi_t), \quad (2.3)$$

where  $\{\eta_t\}_{t=1}^\infty$  is a sequence of carefully chosen step sizes and  $G(\mathbf{x}_t, \xi_t)$  denotes the random gradient at  $\mathbf{x}_t$ . SGD is known to achieve the optimal regret order for convex and strongly convex functions and its simplicity also makes it a go to choice for practitioners. Over the years, numerous variants of SGD with improved theoretical and empirical performance have since been developed and their performance analyzed under various measures (See [241, 38] for recent surveys).

The high cost in computing full gradients in large-scale high-dimensional problems and the resistance of SGD to parallel and distributed implementation have prompted the search for alternative approaches that enjoy better scalability and parallelizability.

A natural choice is iterative coordinate minimization (CM) that has been widely used and analyzed for optimizing a known deterministic function [330]. Also known as alternating minimization, CM is rooted in the methodology of decomposing high-dimensional problems into a sequence of simpler low-dimensional ones. Specifically, CM-based algorithms approach the global mini-

minizer by moving successively to the minimizer in each coordinate<sup>1</sup> while keeping other coordinates fixed to their most recent values. For known deterministic objective functions, it is often assumed that the minimizer in each coordinate can be computed, and hence attained in each iteration.

When coordinate-wise minimization is difficult to carry out, coordinate (gradient) descent (CD) can be employed, which takes a single step (or a fixed number of steps) of (gradient) descent along one coordinate and then moves to the next coordinate<sup>2</sup>. For quadratic objective functions, CD with properly chosen step sizes essentially carries out coordinate minimization. For general objective functions, however, it is commonly observed that CM outperforms CD [330, 30, 290].

While CD/CM-based algorithms have been extensively studied for optimizing deterministic functions, their extensions and resulting performance for stochastic optimization are much less explored. CD can be applied to stochastic optimization with little modification. Since the noisy partial gradient along a coordinate can be viewed as an estimate of the full gradient, stochastic coordinate descent (SCD) has little conceptual difference from SGD. In particular, when the coordinate is chosen uniformly at random at each time, the noisy partial gradient along the randomly chosen coordinate is an unbiased estimate of the full gradient. All analyses of the performance of SGD directly apply. More sophisticated extensions of CD-based methods have been developed in a couple of recent studies (see Section 2.1.4).

---

<sup>1</sup>We use the term “coordinate” to also refer to a block of coordinates.

<sup>2</sup>The term coordinate descent is often used to include coordinate minimization. We make an explicit distinction between CD and CM in this paper. The former refers to taking a single step (or a pre-fixed number of steps) of (gradient) descent along one coordinate and then move to another coordinate. The latter moves along each coordinate with the specific goal of arriving at the minimizer (or a small neighborhood) in this coordinate before switching to another coordinate.

Since exact minimization along a coordinate is impossible due to the unknown and stochastic nature of the objective function, the extension of CM to stochastic optimization is much less clear. This appears to be a direction that has not been taken in the literature and is the focus of this work.

### 2.1.3 Main Results

While both CD- and CM-based methods enjoy scalability and parallelizability, CM often offers better empirical performance and has a much lower overhead in message exchange in distributed computing (due to its sublinear order of switching across coordinates in comparison to the linear order in CD). It is thus desirable to extend these advantages of CM to stochastic optimization.

In this paper, we study stochastic coordinate minimization (SCM) for stochastic convex optimization. We develop a general framework for extending any given low-dimensional optimization algorithm to high-dimensional problems while preserving its level of consistency and regret order. Given that exact minimization along coordinates is impossible, the crux of the proposed framework—referred to as Progressive Coordinate Minimization (PCM)—is an optimal control of the minimization precision in each iteration. Specifically, a PCM algorithm is given by a tuple  $(\{\epsilon_k\}, v, \tau)$ , where  $\{\epsilon_k\}_{k \in \mathbb{N}}$  governs the progressive precision of each CM iteration indexed by  $k$ ,  $v$  is an arbitrary low-dimensional optimization routine employed for coordinate minimization, and  $\tau$  is the self-termination rule for stopping  $v$  at the given precision  $\epsilon_k$  in each iteration  $k$ . We establish the optimal precision control and the resulting order-optimal regret performance for strongly convex and separably non-smooth

functions. An interesting finding is that the optimal progression of precision across iterations is independent of the low-dimensional routine  $v$ , suggesting the generality of the framework for extending low-dimensional optimization algorithms to high-dimensional problems.

We also illustrate the construction of order-optimal termination rules for two specific optimization routines: SGD (applied to minimize along a coordinate) and RWT (recently proposed in [304, 306]). While SGD is directly applicable to high-dimensional problems, its extension within the PCM framework leads to a marriage between the efficiency of SGD with the scalability and parallelizability of CM. RWT as proposed in [304, 306] is only applicable to one-dimensional problems. With no hyper-parameters to tune, however, it has an edge over SGD in terms of robustness and self-adaptivity to unknown function characteristics. For both low-dimensional routines, we demonstrate their high-dimensional extensions within the PCM framework. Empirical experiments using the MNIST dataset show superior performance of PCM over SCD, which echoes the comparison between CM and CD in deterministic settings.

#### 2.1.4 Related Work

CD/CM-based methods for optimizing a known deterministic function have a long history. While such methods had often been eclipsed by more high-performing algorithms, they have started to enjoy increasing popularity in recent years due to the shifted needs from high accuracy to low cost, scalability, and parallelizability in modern machine learning and data analytics applications [140, 139, 215, 216, 236, 234]. See [330, 108] for a detailed literature survey

with insights on the development of CD/CM methods over the years.

Early studies on the convergence of CM-based approaches include [201, 293, 294, 295, 245]. CD-based methods have proven to be easier to analyze, especially under the setup of randomized selection of coordinates [215, 184, 288, 286, 90, 265, 76, 159, 246]. Such CD/CM-based algorithms are often referred to as stochastic CD/CM in the literature due to the randomly chosen coordinates. Optimizing a known deterministic function, however, they are fundamentally different from the stochastic optimization algorithms considered in this work. The term CD/CM with random coordinate selection as used in [233, 200] gives a more accurate description.

CD-based methods have been extended to mini batch settings or for general stochastic optimization problems [229, 314, 343, 79, 231, 331, 336, 170]. In particular, Dang and Lan [79] extended block mirror descent to stochastic optimization. Relying on an averaging of decision points over the entire horizon to combat stochasticity, this algorithm is not applicable to online settings and does not seem to render tractable regret analysis. Wang and Banerjee [314] gave an online implementation of SCD, which we compare with in Section 2.7.

The progressive precision control framework developed in this work bears similarity with inexact coordinate minimization that has been studied in the deterministic setting (see, for example, [90, 229, 118, 287]). The motivation for inexact minimization in these studies is to reduce the complexity of the one-dimensional optimization problem, which is fundamentally different from the root cause arising from the unknown and stochastic nature of the objective function. The techniques involved hence are inherently different with different design criteria.

## 2.2 Problem Formulation

We consider first-order stochastic convex optimization with bandit feedback. The objective function  $f(\mathbf{x})$  over a convex and compact set  $\mathcal{X} \subset \mathbb{R}^d$  is unknown and stochastic as given in Eqn. (2.1). Let  $g(\mathbf{x}) \equiv \nabla f(\mathbf{x})$  be the (sub)gradient of  $f(\mathbf{x})$ . Let  $G(\mathbf{x}; \xi)$  denote unbiased gradient estimates satisfying  $\mathbb{E}_\xi[G(\mathbf{x}; \xi)] = g(\mathbf{x})$ . Let  $g_i(\mathbf{x})$  (similarly,  $G_i(\mathbf{x}; \xi)$ ) denote the partial (random) gradient along the  $i$ -th coordinate ( $i = 1, \dots, d$ ). Let  $\mathbf{x}_i$  and  $\mathbf{x}_{-i}$  denote, respectively, the  $i$ -th element and the  $(d - 1)$  elements other than the  $i$ -th element of  $\mathbf{x}$ . We point out that while we focus on coordinate-wise decomposition of the function domain, extension to a general block structure is straightforward.

### 2.2.1 The Objective Function

We consider objective functions that are convex and possibly non-smooth with the following composite form:

$$f(\mathbf{x}) = \psi(\mathbf{x}) + \phi(\mathbf{x}), \quad (2.4)$$

where  $\phi(\mathbf{x})$  is a coordinate-wise separable convex function (possibly non-smooth) of the form  $\phi(\mathbf{x}) = \sum_{i=1}^d \phi_i(\mathbf{x}_i)$  for some one-dimensional functions  $\{\phi_i(x), x \in \mathcal{X}_i\}_{i=1}^d$  and  $\psi$  is  $\alpha$ -strongly convex and  $\beta$ -smooth. More specifically, for all  $\mathbf{x}, \mathbf{y} \in \mathcal{X}$

$$\psi(\mathbf{y}) \geq \psi(\mathbf{x}) + \langle \nabla \psi(\mathbf{x}), \mathbf{y} - \mathbf{x}, + \rangle \frac{\alpha}{2} \|\mathbf{y} - \mathbf{x}\|_2^2 \quad (2.5)$$

$$\|\nabla \psi(\mathbf{x}) - \nabla \psi(\mathbf{y})\| \leq \beta \|\mathbf{x} - \mathbf{y}\|_2 \quad (2.6)$$

Let  $\mathcal{F}_{\alpha, \beta}$  denote the set of all such functions.

The above composite form of the objective function has been widely adopted in the literature on CM and CD [330]. The separably non-smooth component  $\phi$  arises naturally in many machine learning problems that often involve separable regularization such as  $\ell_1$  norm and box constraints.

## 2.2.2 Regret, Consistency, and Efficiency Measures

A stochastic optimization algorithm  $\Upsilon = \{\Upsilon_t\}_{t=1}^T$  is a sequence of mappings from past actions and observations to the next choice of query point. The performance of  $\Upsilon$  is measured by the cumulative regret defined as

$$R_\Upsilon(T) = \mathbb{E} \left[ \sum_{t=1}^T F(\mathbf{x}_t, \xi_t) - F(\mathbf{x}^*, \xi_t) \right] \quad (2.7)$$

where the expectation is with respect to the random process of the query points and gradient observations induced by the algorithm  $\Upsilon$  under the i.i.d. endogenous process of  $\{\xi_t\}_{t=1}^T$ .

In general, the performance of an algorithm depends on the underlying unknown objective function  $f$  (a dependency omitted in the regret notation for simplicity). Consider, for example, an algorithm that simply chooses one function in  $\mathcal{F}_{\alpha,\beta}$  and sets its query points  $\mathbf{x}_t$  to the minimizer of this function for all  $t$  would perform perfectly for the chosen function but suffers a linear regret order for all objective functions with sufficient deviation from the chosen one. It goes without saying that such heavily biased algorithms that completely forgo learning are of little interest.

We are interested in algorithms that offer good performance for all functions in  $\mathcal{F}_{\alpha,\beta}$ . An algorithm  $\Upsilon$  is *consistent* if for all  $f \in \mathcal{F}_{\alpha,\beta}$ , the end point  $\mathbf{x}_T$  produced

by  $\Upsilon$  satisfies

$$\lim_{T \rightarrow \infty} \mathbb{E}[f(\mathbf{x}_T)] = f(\mathbf{x}^*). \quad (2.8)$$

A consistent algorithm offers a sublinear regret order. This is also known as Hannan consistency or no-regret learning [124]. The latter term makes explicit the diminishing behavior of the average regret per action.

To measure the convergence rate of an algorithm, we introduce the concept of *p-consistency*. For a parameter  $p \in (0, 1)$ , we say  $\Upsilon$  is *p-consistent* if

$$\sup_{f \in \mathcal{F}_{\alpha,\beta}} (\mathbb{E}[f(\mathbf{x}_T)] - f(\mathbf{x}^*)) \sim \Theta(T^{-p}). \quad (2.9)$$

A *p*-consistent algorithm offers an  $O(T^{1-p})$  regret order for all  $f \in \mathcal{F}_{\alpha,\beta}$ . The parameter  $p$  measures the convergence rate.

An *efficient* algorithm is one that achieves the optimal convergence rate, hence lowest regret order. Specifically,  $\Upsilon$  is *efficient* if for all initial query points  $\mathbf{x}^{(1)} \in \mathcal{X}$ , the end point  $\mathbf{x}_T$  produced by  $\Upsilon$  satisfies, for some  $\lambda > 0$ ,

$$\sup_{f \in \mathcal{F}_{\alpha,\beta}} (\mathbb{E}[f(\mathbf{x}_T)] - f(\mathbf{x}^*)) \sim (f(\mathbf{x}^{(1)}) - f(\mathbf{x}^*))^\lambda \Theta(T^{-1}). \quad (2.10)$$

An efficient algorithm offers the optimal  $\log T$  regret order for all  $f \in \mathcal{F}_{\alpha,\beta}$ . In addition, it is able to leverage favorable initial conditions when they occur. We note here that the specific value of  $\lambda$  affects only the leading constant, but not the regret order. Hence for simplicity, we often use *p*-consistency with  $p = 1$  to refer to efficient algorithms.

## 2.3 Progressive Coordinate Minimization

In this section, we present the PCM framework for extending low-dimensional optimization routines to high-dimensional problems. After specifying the gen-

eral structure of PCM, we lay out the optimality criteria for designing its constituent components.

### 2.3.1 The General Structure of PCM

Within the PCM framework, an algorithm is given by a tuple  $\Upsilon(\{\epsilon_k\}, \nu, \tau)$ , where  $\{\epsilon_k\}_{k \in \mathbb{N}}$  governs the progressive precision of each CM iteration indexed by  $k$ ,  $\nu$  is the low-dimensional optimization routine employed for coordinate minimization, and  $\tau$  is the self-termination rule (i.e., a stopping time) for stopping  $\nu$  at the given precision  $\epsilon_k$  in each iteration  $k$ . Let  $\tau(\epsilon)$  denote the (random) stopping time for achieving  $\epsilon$ -precision under the termination rule  $\tau$ .

A PCM algorithm  $\Upsilon(\{\epsilon_k\}, \nu, \tau)$  operates as follows. At  $t = 1$ , an initial query point  $\mathbf{x}^{(1)}$  and coordinate  $i_1$  are chosen at random. The CM routine  $\nu$  is then carried out along coordinate  $i_1$  with all other coordinates fixed at  $\mathbf{x}_{-i_1}^{(1)}$ . At time  $\tau(\epsilon_1)$ , the first CM iteration ends and returns its last query point  $\mathbf{x}_{\tau(\epsilon_1), i_1}$ . The second iteration starts along a coordinate  $i_2$  chosen uniformly at random and with the  $i_1$  coordinate updated to its new value  $\mathbf{x}_{\tau(\epsilon_1), i_1}$ . The process repeats until the end of horizon  $T$  (see Algorithm 1 below).

### 2.3.2 Optimal Design of Constituent Components

PCM presents a framework for extending low-dimensional optimization algorithms to high-dimensional problems. The CM routine  $\nu$  in a PCM algorithm is thus given, and we allow it to be an arbitrary  $p$ -consistent algorithm for any  $p \in (0, 1]$  (note that the definitions of  $p$ -consistency and efficiency in Section 2.2

---

**Algorithm 1** PCM  $\Upsilon(\{\epsilon_k\}, \nu, \tau)$ 

---

**Input:** initial point  $\mathbf{x}^{(1)}$ .  
Set  $k \leftarrow 1, t \leftarrow 1$   
**repeat**  
    Choose coordinate  $i_k$  uniformly at random.  
    Carry out  $\nu$  along the direction  $i_k$  as follows:  
        Set the initial point to  $\mathbf{x}_{i_k}^{(k)}$  with fixed  $\mathbf{x}_{-i_k}^{(k)}$ .  
        Continue until  $\tau(\epsilon_k)$ .  
        Return the final point  $\mathbf{x}_{\tau(\epsilon_k), i_k}$ .  
     $\mathbf{x}^{(k+1)} \leftarrow (\mathbf{x}_{\tau(\epsilon_k), i_k}, \mathbf{x}_{-i_k}^{(k)})$   
     $k \leftarrow k + 1$   
     $t \leftarrow t + \tau(\epsilon_k)$   
**until**  $t = T$

---

apply to arbitrary dimension.) Allowing arbitrary low-dimensional routines make PCM generally applicable, and the inclusion of consistent but not efficient (i.e.,  $p < 1$ ) routines responds to the shifted needs for low-cost solutions of only modest accuracy, as seen in modern machine learning and data analytics applications.

It is readily seen that for every  $f(\mathbf{x}) \in \mathcal{F}_{\alpha, \beta}$ , its low-dimensional restriction  $f(\cdot, \mathbf{x}_{-i})$  for arbitrarily fixed  $\mathbf{x}_{-i}$  belongs in  $\mathcal{F}_{\alpha, \beta}$ . Consequently, for a given low-dimensional routine  $\nu$  with a certain consistency/efficiency level  $p \in (0, 1]$  (which needs to hold for all low-dimensional restrictions in  $\mathcal{F}_{\alpha, \beta}$ ; see (2.9), (2.10)), its high-dimensional extension cannot have a better consistency level (or equivalently, lower regret order). The best possible outcome is that the high-dimensional extension preserves the  $p$ -consistency and the regret order of the low-dimensional algorithm for high-dimensional problems.

The design objective of PCM is thus to choose  $\{\epsilon_k\}$  and  $\tau$  for a given low-dimensional  $p$ -consistent algorithm  $\nu$  such that the resulting high-dimensional algorithm preserves the regret order of  $\nu$ .

The above optimization can be decoupled into two steps. First, the termination rule  $\tau$  is designed to meet an order-optimal criterion as specified below. The optimal design of  $\tau$  is specific to the routine  $v$ , as one would expect. In the second step, the progression of precision  $\{\epsilon_k\}$  is optimized for the given  $v$  augmented with the order-optimal  $\tau$  to preserve the  $p$ -consistency. Quite surprisingly, as shown in Section 2.4, there exists a *universal* optimal  $\{\epsilon_k\}$  that is independent of not only the specific routine  $v$  but also the specific consistency value  $p \in (0, 1]$ .

**Definition 2.3.1.** For a given  $p$ -consistent ( $p \in (0, 1]$ ) low-dimensional algorithm  $v$  and a given  $\epsilon > 0$ , let  $\tau(\epsilon)$  denote a stopping time over the random process of  $\{x_t\}_{t \geq 1}$  induced by  $v$  that satisfies  $\mathbb{E}[f(x_{\tau(\epsilon)})] - f(x^*) \leq \epsilon$ . A termination rule  $\tau$  is order optimal in  $\epsilon$  if

$$\sup_{f \in \mathcal{F}_{\alpha,\beta}} \mathbb{E}[\tau(\epsilon)] \sim \Theta(\epsilon^{-1/p}). \quad (2.11)$$

Note that the above definition is for the dimensionality as determined by the given algorithm  $v$  with  $f$  and  $\mathcal{F}_{\alpha,\beta}$  defined accordingly.

An order-optimal termination rule is one that realizes the exponent  $p$  of the consistency of the underlying algorithm  $v$ . The design of such termination rules is specific to  $v$ , which we illustrate in Section 2.5 for two representative efficient (i.e.,  $p = 1$ ) low-dimensional routines.

## 2.4 The Optimal Precision Control

The theorem below establishes the optimal design of  $\{\epsilon_k\}$  for arbitrary  $p$ -consistent low-dimensional routines.

**Theorem 2.4.1.** *Let  $v$  be an arbitrary  $p$ -consistent ( $p \in (0, 1]$ ) low-dimensional routine and  $\tau$  an order-optimal termination rule. For all  $\gamma \in [(1 - \alpha/(d\beta))^{1/2}, 1)$  and  $\epsilon_0 > 0$ , the PCM algorithm  $\Upsilon(\{\epsilon_0\gamma^k\}, v, \tau)$  achieves a regret of  $O(T^{1-p} \log^p T)$  for all  $f \in \mathcal{F}_{\alpha, \beta}$ .*

Theorem 2.4.1 shows that setting  $\epsilon_k = \epsilon_0\gamma^k$  preserves the regret order<sup>3</sup> of  $v$ . It is thus optimal. Such a choice of  $\{\epsilon_k\}$  is independent of  $v$  as well as the consistency level  $p$  of  $v$ .

The proof of Theorem 2.4.1 is based on a decomposition of the regret as given below. Let  $K$  denote the (random) number of iterations until time  $T$ . Let  $\mathbf{x}_{i_k}^* = \arg \min_x f((\mathbf{x}_{-i_k}^{(k-1)}, x))$  be the minimizer in the  $i_k^{\text{th}}$  coordinate with other coordinates fixed to  $\mathbf{x}_{-i_k}^{(k-1)}$  (i.e., values from the previous iteration). Let  $\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^* = (\mathbf{x}_{-i_k}^{(k-1)}, \mathbf{x}_{i_k}^*)$ . Let  $t_k = t_{k-1} + \tau_v(\epsilon_k)$  with  $t_0 = 0$  denote the (random) time instants that mark the end of each iteration. We then have

$$\begin{aligned} R_\Upsilon(T) &= \mathbb{E} \left[ \sum_{t=1}^T F(\mathbf{x}_t, \xi_t) - F(\mathbf{x}^*, \xi_t) \right] \\ &= \mathbb{E} \left[ \sum_{k=1}^K \sum_{t=t_{k-1}+1}^{t_k} F(\mathbf{x}_t, \xi_t) - F(\mathbf{x}^*, \xi_t) \right]. \end{aligned}$$

---

<sup>3</sup>The preservation of the regret order is exact for efficient routines. For consistent but not efficient (i.e.,  $p < 1$ ) routines, the preservation is up to a poly-log term which is dominated by the term of  $T^{1-p}$ .

This can be split into two terms using the local minimizer as

$$R_\Upsilon(T) = \mathbb{E} \left[ \underbrace{\sum_{k=1}^K \sum_{t=t_{k-1}+1}^{t_k} [F(\mathbf{x}_t, \xi_t) - F(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*)]}_{R_1} \right] + \mathbb{E} \left[ \underbrace{\sum_{k=1}^K \sum_{t=t_{k-1}+1}^{t_k} [F(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*) - F(\mathbf{x}^*)]}_{R_2} \right]. \quad (2.12)$$

The first term  $R_1$  corresponds to the regret incurred by the low-dimensional routine  $\nu$  carried out along one dimension. Note that this regret is computed with respect to the one-dimensional local minima  $\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*$ . The second term  $R_2$  corresponds to the loss incurred at the one-dimensional local minima in excess to the global minimum  $\mathbf{x}^*$ .

The above regret decomposition also provides insight into the optimal design of  $\{\epsilon_k\}$ . To achieve a low regret order, it is desirable to equalize the orders of  $R_1$  and  $R_2$ . If a more aggressive choice of  $\epsilon_k$  is used, then the rate of decay of the CM iterates is unable to compensate for the time required for higher accuracy, resulting in  $R_2$  dominating  $R_1$ . On the other hand, a more conservative choice will lead to a slower decay in objective function with an increased number of CM iterations. This would result in increasing both the terms to an extent where  $\Upsilon$  will no longer be able to maintain the consistency level of  $\nu$ .

*Proof.* We give here a sketch of the proof. The analysis of  $R_1$  and  $R_2$  builds on analytical characterizations of the following two key quantities: the expected number  $\mathbb{E}[K]$  of CM iterations and the convergence rate of CM outputs  $\{\mathbf{x}^{(k)}\}_{k=1}^K$ . They are given in the following two lemmas.

**Lemma 2.4.2.** *Let  $\nu$  be a  $p$ -consistent policy for some  $p \in (0, 1]$  and  $\tau$  its order-optimal termination rule. Under  $\Upsilon(\{\epsilon_0 \gamma^k\}, \nu, \tau)$ , we have  $\mathbb{E}[K] \sim O(\log T)$  for all  $f \in \mathcal{F}_{\alpha, \beta}$ .*

**Lemma 2.4.3.** *Let  $\{\mathbf{x}^{(k)}\}$  be the sequence of CM outputs generated under  $\Upsilon(\{\epsilon_0 \gamma^k\}, \nu, \tau)$*

for a function  $f \in \mathcal{F}_{\alpha,\beta}$ . Then the sequence of points  $\{\mathbf{x}^{(k)}\}$  satisfy  $\mathbb{E}[f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*)] \leq F_0\gamma^k$  for all  $k \geq 0$  and for all  $\gamma \in [(1 - \alpha/(d\beta))^{1/2}, 1)$  where  $F_0 = \max\{f(\mathbf{x}_0) - f(\mathbf{x}^*), \epsilon_0/(1 - \gamma)\}$ .

$R_1$  is bounded using the consistency level of the routine  $v$  augmented with the termination rule  $\tau$ .  $R_2$  is bounded using Lemma 2.4.3 and the expected time taken in each CM iteration. On plugging in the value of  $\epsilon_k$ , both terms end up being of the same order and we arrive at the theorem. The detailed proofs of the lemmas and the theorem can be found in Appendix A.1.  $\square$

## 2.5 Termination Rules

In this section, we illustrate the construction of order-optimal termination rules for two representative and fundamentally different low-dimensional routines, one classical, one recent. For simplicity, we focus on smooth objective functions. All notations are for a specific coordinate with coordinate index omitted.

### 2.5.1 SGD

For a given initial point  $x_1$ , SGD proceeds by generating the following sequence of query points

$$x_{t+1} = \text{proj}_{\mathcal{X}}(x_t - \eta_t G(x_t, \xi_t)), \quad (2.13)$$

where  $G(x_t, \xi_t)$  is the random gradient observed at  $x_t$ ,  $\{\eta_t\}_{t \geq 1}$  is the sequence of step sizes, and  $\text{proj}_{\mathcal{X}}$  denotes the projection operator onto the convex set  $\mathcal{X}$  (restricted to the chosen coordinate with other coordinates fixed). The following

lemma establishes the efficiency of the SGD routine with properly chosen hyperparameters and the order optimality of the termination rule for noise models with bounded variance. Based on Theorem 2.4.1, we can then conclude that the resulting PCM algorithm with  $\{\epsilon_k\} = \epsilon_0 \gamma^k$  is an efficient algorithm with a regret of  $O(\log T)$ .

**Lemma 2.5.1.** *Consider the low-dimensional routine of SGD with step sizes given by  $\eta_t = \mu/(1 + vt)$  with  $\mu = \frac{\mu_0 \alpha}{2g_{\max}^2}$  and  $v = \frac{\mu_0 \alpha^2}{4g_{\max}^2}$ , where  $g_{\max}$  is an upper bound on the second moment of the random gradient,  $G(x, \xi)$ , for all  $x \in \mathcal{X}$  and  $\mu_0$  a properly chosen hyperparameter. Then SGD with the above chosen parameters is an efficient algorithm as defined in (2.10). The termination rule given by  $\tau(\epsilon) = \left\lceil \frac{2\beta g_{\max}^2}{\alpha^2 \epsilon} \right\rceil$  is order optimal as defined in Definition 2.3.1.*

*Proof.* We give here a sketch of the proof. Details can be found in the supplementary material. The order-optimality of the termination rule follows immediately from definition 2.3.1. For implementation in PCM, the constant  $\mu_0$  is set to  $\mu_0 \sim \Theta(\gamma^k)$  in iteration  $k$  to ensure the adaptivity to the initial point. Using smoothness of  $f$ , we obtain  $\mathbb{E}[f(x_t) - f(x^*)] \leq \beta \mathbb{E}[|x_t - x^*|^2] \leq \mu_0/(1 + vt)$ , implying that SGD is a consistent algorithm with  $p = 1$ . The choice of  $\mu_0$  makes it an efficient algorithm with  $\lambda = 1$ .  $\square$

## 2.5.2 RWT

RWT (Random Walk on a Tree) proposed in [304, 306] is restricted to one-dimensional problems. There do not appear to be any simple extensions of RWT to high-dimensional problems. We show here that PCM offers a possibility and preserves its efficiency. We first describe the original RWT algorithm and then

describe its extension to high dimensional settings within the PCM framework. We conclude the section by stating the regret guarantees of the proposed algorithm.

### 2.5.3 The RWT algorithm

Without loss of generality, assume that the one-dimensional domain is the closed interval  $[0, 1]$ . With a slight abuse of notation, let  $F(x, \xi)$  be the one dimensional stochastic function to be minimized and  $G(x, \xi)$  denote its stochastic gradient while  $f(x)$  and  $g(x)$  respectively denote their expected values. Also we assume that  $|g(x)| \leq g_{\max}$  for all  $x \in \mathcal{X}'$ , where  $\mathcal{X}'$  is the domain of the function.

The basic idea of RWT is to construct an infinite-depth binary tree based on successive partitions of the interval. Each node of the tree represents a sub-interval, and nodes on the same level give an equal-length partition of  $[0, 1]$ . The query point at each time is then generated based on a biased random walk on the interval tree that initiates at the root and is biased toward the infinitesimally small interval containing the minimizer  $x^*$ . When the random walk reaches a node, the two end points along with the middle point of the corresponding interval are queried in serial to determine, with a required confidence level  $\check{p}$ , the sign of  $g(x)$  at those points. The test on the sign of  $g(x)$  at any given  $x$  is done through a confidence-bound based local sequential test using random gradient observations. The outcomes of the sign tests at the three points of the interval determines the next move of the random walk: to the child that contains a sign change or back to the parent under inconsistent test outcomes. The confidence level  $\check{p}$  of the sign test ensures the bias of the walk, i.e., the probability of moving

toward  $x^*$  is greater than  $1/2$  at each step. For one-dimensional problems, the biased walk on the tree continues until  $T$ .

A crucial aspect of the above algorithm is the local sequential test. Let the sample mean of  $s$  samples of the stochastic gradient at a point  $x \in \mathcal{X}'$  be denoted as  $\bar{G}_s(x) = \frac{1}{s} \sum_{t=1}^s G(x, \xi_t)$ . The sequential test in RWT for sub-Gaussian noise is given below. For heavy-tailed noise, the only required change to RWT is in the confidence bounds used in the sequential test (see [304]).

- ▷ If  $\bar{G}_s(x) > \sqrt{\frac{5\sigma_0^2}{s} \log\left(\frac{6 \log s}{\sqrt{\check{p}}}\right)}$ , terminate; output 1.
- ▷ If  $\bar{G}_s(x) < -\sqrt{\frac{5\sigma_0^2}{s} \log\left(\frac{6 \log s}{\sqrt{\check{p}}}\right)}$ , terminate; output  $-1$ .
- ▷ Otherwise, take another sample of  $G(x, \xi)$  and repeat.

Figure 2.1: The sequential test at a sampling point  $x$  under sub-Gaussian noise.

where  $\check{p}$  is the confidence parameter for the sequential test. To ensure the bias in the random walk,  $\check{p}$  is set to a value in  $(0, 1 - 2^{-1/3})$ .

#### 2.5.4 RWT Algorithm for PCM

In the high-dimensional extension of RWT using PCM, we use RWT equipped with a termination rule as the low dimensional optimization routine. Specifically, in the  $k^{\text{th}}$  iteration of PCM, consider the interval, i.e., one dimensional domain, given by  $\{x : (x, \mathbf{x}_{-i_k}^{(k-1)}) \in \mathcal{X}\}$ , that is, all the points in the domain whose all but the  $i_k^{\text{th}}$  coordinates are same as that of  $\mathbf{x}^{(k-1)}$ . This corresponds to the do-

main for which we use the RWT algorithm described in the previous section. However, for this high-dimensional setting we make two minor modifications to the original algorithm, including the addition of a termination rule.

Firstly, note that in Section 2.5.3, the random walk is initialized at the root of the tree as there is no prior information about the location of the minimizer. However, if we have some prior information about the location of the minimizer, we can initialize the random walk at a lower level in the tree. This enables us to give higher preference to the region where the minimizer is likely to be located, thereby reducing the expected time to convergence. Consequently, such an initialization allows RWT to leverage favorable initial conditions. Therefore, for PCM, we initialize RWT at the node which contains the initial point and is at a level where the interval length is lesser than  $\sqrt{\log_2\left(\frac{\beta\sqrt{2}}{\sqrt{\alpha}\epsilon}\right)\frac{2\mu_0}{\alpha}}$ , where  $\mu_0$  is a carefully chosen hyperparameter and  $\epsilon$  is the required precision. If the threshold exceeds 1, then we begin at the root. The significance of this choice of initialization and the allowed values of  $\mu_0$  are discussed in Appendix A.3.2.

Secondly, we equip RWT with the following termination rule that leverages the structure of the local confidence-bound based sequential test in RWT (See Fig. 2.1). Note that the precision condition required at termination can be translated to an upper bound on the magnitude of the gradient. Since the local test is designed to estimate the sign of the gradient, it naturally requires more samples as the gradient gets closer to zero (i.e., the signal strength reduces while the noise persists). Assume that the noise is sub-Gaussian, that is, the moment generating function of  $G_i(\mathbf{x}; \xi) - g_i(\mathbf{x})$  is upper bounded by that of a Gaussian with variance  $\sigma_i^2$  for  $i = 1, 2, \dots, d$ . We propose the following termination rule: the current CM iteration terminates once a sequential test draws more than

$$N_0(\epsilon) = \frac{40\sigma_0^2}{\alpha\epsilon} \log\left(\frac{2}{\check{p}} \log\left(\frac{80\sigma_0^2}{\alpha\check{p}\epsilon}\right)\right) \text{ samples, where } \sigma_0^2 \geq \max_i \sigma_i^2.$$

This threshold is so designed that when the number of samples in the sequential test exceeds that value, the gradient at that point is sufficiently small with high probability, leading to the required precision. It is interesting to note that the termination rule for SGD given in Lemma 2.5.1 is an open-loop design with pre-fixed termination time, while the termination rule proposed for RWT is closed-loop and adapts to random observations.

The following lemma gives the regret order of the high-dimensional extension of RWT within the PCM framework. The detailed proof of the lemma is can be found in Appendix A.3.4.

**Lemma 2.5.2.** *The PCM-RWT algorithm with the chosen termination rule has a regret order of  $O(\log T(\log \log T)^2)$ .*

## 2.6 Discussions and Extensions

**Parallel and Distributed Computing:** One of the major advantages of CD/CM based methods is their amenability to parallel and distributed computing, which PCM naturally inherits. Advantages of CD/CM methods in parallel and distributed computing have been well studied in the literature [236, 234, 42, 223, 197, 205, 235]. It has been shown that the convergence of coordinate-based methods with parallel updates is guaranteed only when the parallel updates are aggregated in such a way that the combined update leads to a decrease in the function value as compared to the previous iterate. Such a condition is possible to verify and enforce when the objective function is deter-

ministic and known, but presents a challenge in stochastic optimization.

To achieve parallelization of PCM that maintains the  $p$ -consistency as in the serial implementation, we draw inspiration from the technique proposed in Ferris and Mangasarian [109] that leverages the convexity of the objective function.

Consider the setup of  $m < d$  cores, connected in parallel to the main server. To make it similar to our original setup, we assume that each processor has the access to the oracle independently of others. The algorithm for implementing PCM using parallel updates is described as follows:

1. Read the current iterate  $\mathbf{x}$  and pass it to all cores.
2. Select  $m$  different indices from  $\{1, 2, \dots, d\}$  uniformly at random and allocate them to the cores.
3. On each core, run the one dimensional optimization routine along the dimension whose was index assigned to that core. The initial point for all the cores will be the same point  $\mathbf{x}$ . Let the points returned by the cores to the server be denoted as  $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m$ .
4. Generate the next iterate  $\mathbf{x}_1 = \frac{1}{m} \sum_{k=1}^m \mathbf{y}_k$ .

The last step is the update or the synchronization step which ensures that the function value at the new iterate is lesser than that at the one previous one. Note that in the second step  $m$  different indices are chosen uniformly at random, that is, one of the  $\binom{d}{m}$  sets is chosen. It can be shown that with a parallel implementation of PCM, the “effective” dimension of the problem is reduced from  $d$  to  $d/m$ , offering a linear speedup in the number of devices. Please refer to Appendix A.4 for a proof of this claim.

**Heavy-Tailed Noise:** The noise in the partial gradient estimates affects only the consistency level  $p$  of the low-dimensional routine  $v$ , subsequently, the design of the order-optimal termination rule. The optimal precision control, however, is independent of the noise characteristics. More specifically, the same optimal precision control as specified in Theorem 2.4.1 preserves the  $p$ -consistency and regret order of the low-dimensional routine  $v$  even under heavy-tailed noise. In particular, for heavy-tailed noise with a finite  $b^{\text{th}}$  moment ( $b \in (1, 2)$ ), both SGD and RWT are  $(2 - 2/b)$ -consistent and offer an optimal regret order of  $O(T^{2/b-1})$  (up to poly-log  $T$  factors) [339, 304]. It is not difficult to show that under heavy-tailed noise, an open-loop termination rule with  $\tau(\epsilon) = C\epsilon^{\frac{b}{2(1-b)}}$  for SGD and a similar sample-threshold based termination rule with  $N_0(\epsilon) = C'\epsilon^{\frac{b}{2(1-b)}} \text{polylog}(1/\epsilon)$  for RWT are order optimal, where  $C$  and  $C'$  are constants. In conclusion, PCM offers an optimal high-dimensional extension for any  $p$ -consistent algorithm regardless of the noise characteristics.

**Zeroth-order feedback model:** The two example low-dimensional stochastic optimization routines used in Section 2.5 are first-order algorithms that use gradient to update the query points. However, the PCM framework is equally applicable to zeroth-order algorithms which directly learn from random losses  $F(x_t; \xi)$ . As it can be seen from Theorem 1, the PCM framework is agnostic to the low-dimensional routine  $v$  and the associated termination rule (provided it is order optimal).

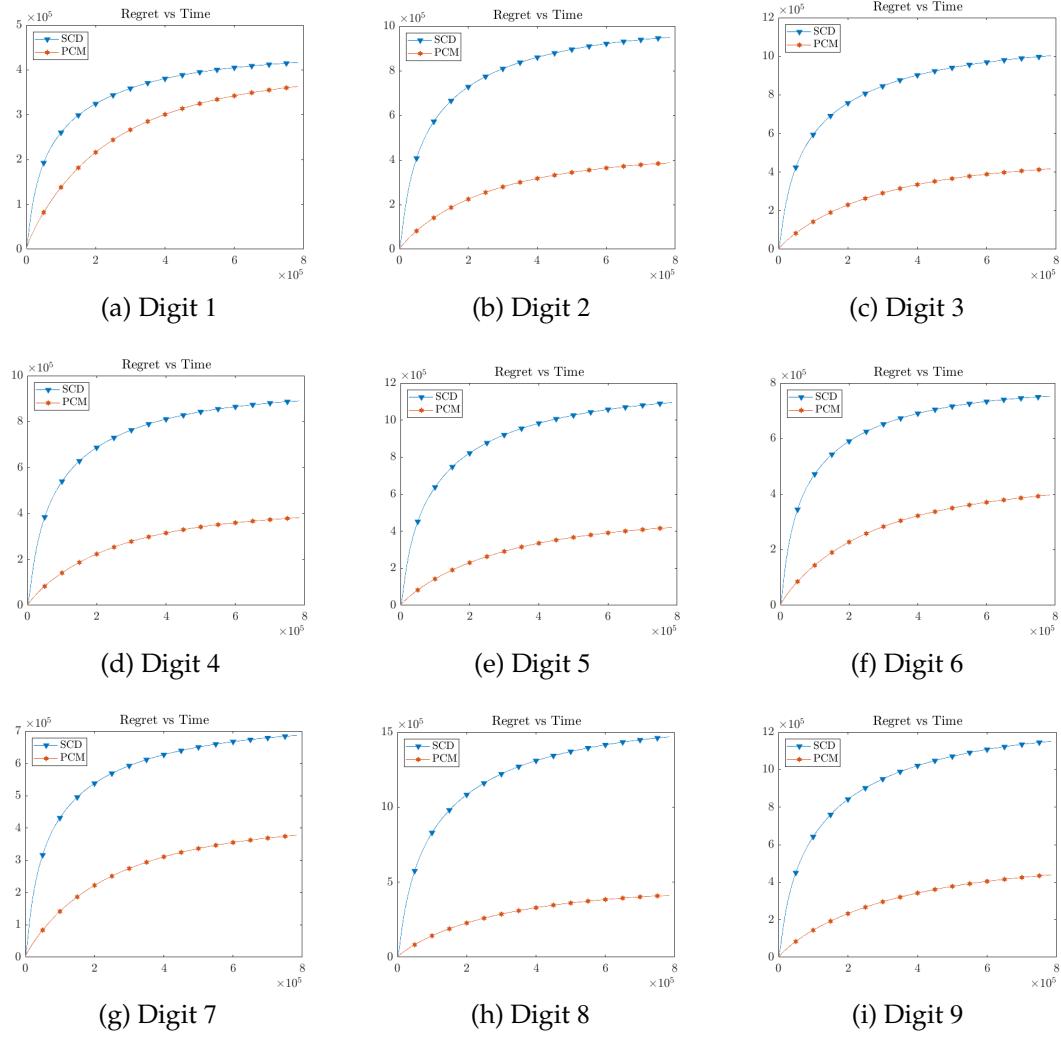


Figure 2.2: Plot of cumulative regret against time for different algorithms for all 9 classification tasks.

## 2.7 Empirical Results

In this section, we compare the regret performance of PCM employing SGD with that of SCD (based on its online version developed in [314]). We consider the problem of binary classification on the MNIST dataset [180] as a one-vs-rest classification problem in an online setting. We use regularized hinge loss as the loss function. At each time  $t$ ,  $\xi_t = (Y_t, Z_t)$  is drawn uniformly at random from the

dataset. Then using the current classifier  $\mathbf{x}_t$ , we incur a random loss

$$F(\mathbf{x}_t, \xi_t) = \max\{0, 1 - Z_t \langle \mathbf{x}_t, Y_t \rangle\} + \frac{\alpha}{2} \|\mathbf{x}_t\|^2 \quad (2.14)$$

and observe the partial gradient given as

$$G_{i_t}(\mathbf{x}_t, \xi_t) = -Z_t(Y_t)_{i_t} \mathbb{1}\{1 - Z_t \langle \mathbf{x}_t, Y_t \rangle > 0\} + \alpha(\mathbf{x}_t)_{i_t} \quad (2.15)$$

where  $i_t$  denotes the index of the coordinate chosen at time  $t$ . Both algorithms are randomly initialised with a point in  $[-0.5, 0.5]^d$  and run for  $T = 1000d$ , where  $d = 785$  is the dimensionality of the problem. The regularization parameter is set to  $\alpha = 1.2 \times 10^{-2}$ . The regret plotted is averaged over 10 Monte Carlo runs. The SCD algorithm is run with stepsize  $\eta_t = 5/\lceil t/10000 \rceil$ . The PCM algorithm is implemented using a SGD as the local optimization routine with  $\gamma = 0.99999$ ,  $\epsilon_0 = 0.1$  and step size of  $\eta = 0.2$ . The step size in each iteration was reduced by a factor of  $\gamma$ . The termination rule was set to  $\lceil 1/2\epsilon_k \rceil$ . The parameters in both algorithms were optimized based on a grid search. The results obtained for various digits are shown in Figure 2.2.

It is evident from the plots in Figure 2.2 that PCM has a significantly better performance. It suggests that the advantages of CM over CD type methods also hold in stochastic optimization.

## CHAPTER 3

### KERNEL-BASED OPTIMIZATION

#### 3.1 Introduction

Consider a typical black-box optimization problem with an unknown objective function  $f : \mathcal{X} \rightarrow \mathbb{R}$ , where  $\mathcal{X} \subset \mathbb{R}^d$  is a convex and compact set. In the previous chapter, we studied the class of problems where unknown function is assumed to satisfy convexity, leading to the class of stochastic convex optimization problems. Another class of black-box optimization problems that is gaining interest in recent years is kernel-based learning where unknown objective function,  $f$ , is assumed to live in a Reproducing Kernel Hilbert Space (RKHS) associated with a positive-definite kernel. An effective approach to kernel-based black-box optimization is Bayesian optimization that adopts a *fictitious* prior on the unknown function  $f$ . In other words, while  $f$  is deterministic, it is viewed internally by the learning algorithm as a realization of a random process over  $\mathcal{X}$ . A natural choice is the Gaussian process (GP) with a Gaussian prior due to the conjugate property that significantly simplifies the analytical form of the posterior distribution at each newly obtained observation.

##### 3.1.1 Gaussian Process Models

In a celebrated work, Srinivas *et al.* [278] proposed the GP-UCB algorithm that constructs a proxy of  $f$  using the upper confidence bound (UCB) concept first introduced in the classical multi-armed bandit problem [175, 15]. Specifically, at each time instant  $t$ , a UCB of  $f$  is constructed using the closed-form posterior

mean and standard deviation of the GP model of  $f$ . The algorithm then sets the next query point to be the maximizer of the UCB. Several variations of GP-UCB, tailored for different settings (see Sec 3.1.3), have since been developed.

The GP-UCB family of algorithms generally enjoy good empirical performance in terms of regret. The analytical guarantees of their regret performance, however, leave considerable gaps to the existing lower bound [257]. More significantly, the state-of-the-art regret bound of GP-UCB does not guarantee a sub-linear order in  $T$  for certain kernels, hence a lack of guaranteed convergence to  $f(x^*)$  [257, 148].

Another difficulty with the GP-UCB family of algorithms is their computational complexity, which can be prohibitive as the dimension  $d$  and/or the horizon length  $T$  grows. The computational complexity has two main sources: (i) the inversion of the covariance matrix in updating the posterior GP distribution, which has an  $O(t^3)$  complexity with  $t$  samples; (ii) the maximization of the UCB proxy over the entire domain  $\mathcal{X}$  at each time instant. In particular, due to the multi-modality of the UCB score, its maximization is often carried out using a grid search with an increasingly finer discretization of the entire domain. Specifically, due to analytical requirements, the discretization is typically assumed to grow in the order of  $O(t^{2d})$  [278, 68], resulting in an overall computational complexity of  $O(T^{2d+3})$ .

Several studies exist that tackle the first source of high complexity of GP-UCB, using sparse matrix approximation techniques to reduce the complexity in the inversion of the covariance matrix (see, e.g., [208, 49]). The second source, which is the dominating factor, has not been effectively addressed.

### 3.1.2 Main results

The goal of this chapter is to develop a GP-based Bayesian Optimization algorithm with a regret guarantee that closes the gap to the lower bound. Furthermore, we tackle the second source of the complexity to ensure both learning efficiency and computational efficiency. Referred to as GP-ThreDS (Thresholded Domain Shrinking), the proposed algorithm is rooted in the methodology of domain shrinking: it continuously prunes sub-performing regions of the domain  $\mathcal{X}$  and zooms into increasingly smaller high-performing regions of  $\mathcal{X}$  as time goes. The purpose of the domain shrinking is twofold. First, it ensures high learning efficiency by focusing queries on regions of  $\mathcal{X}$  with function values approaching  $f(x^*)$ . Second, it achieves computational efficiency by avoiding a global maximization of the proxy function over the entire domain  $\mathcal{X}$ .

Our specific approach to domain shrinking is built upon a sequence of localized searches on a growing binary tree that forms successively refined partitions of  $\mathcal{X}$ . Starting from the root of the tree that represents the entire domain, the search progresses down the tree by adaptively pruning nodes that do not contain the maximizer with high probability, consequently zooming into increasingly smaller high-performing regions of  $\mathcal{X}$  as the search deepens. Another progressive thread in this sequence of localized searches is the criterion for pruning the tree. Each localized search aims to identify nodes at a certain depth of the tree that contain points with function values exceeding a given threshold. The threshold is updated iteratively to approach the maximum function value  $f(x^*)$ . More succinctly, the proposed algorithm is a sequence of localized searches in the domain of the function guided by an iterative search in the range of the function.

The above domain shrinking approach via localized search is the primary contributing factor to improved performance in terms of both regret guarantee and computational complexity. In particular, the rate of domain shrinking is controlled to ensure not only the concentration of query points in high-performing regions, but also a *constant*-sized discretization at all times when estimating the function values. This constant-sized discretization allows a tighter regret analysis and results in a regret upper bound for GP-ThreDS that matches with the lower bound (up to a poly-logarithmic factor). We show that the regret of GP-ThreDS is  $O(\sqrt{T\gamma_T})$  (up to a poly-logarithmic factor), where  $\gamma_T$  denotes the maximum information gain after  $T$  steps and is representative of the *effective* dimension of the problem [340, 308]. In the case of Matérn and Squared Exponential (SE) kernels where the lower bounds on regret are known, on substituting the improved bounds on  $\gamma_T$  from [301], our results match the lower bounds and close the gap reported in [257, 48]. In comparison, the state-of-the-art analysis of GP-UCB yields an  $O(\gamma_T \sqrt{T})$  regret bound (e.g., see [68, Theorem 3]). The  $O(\sqrt{\gamma_T})$  gap between the regret guarantees of GP-UCB and the proposed GP-ThreDS is significant: it can grow polynomially in  $T$  (e.g. in the case of Matérn kernel).

Computation-wise, the constant-sized discretization contrasts sharply with the growing (at rate  $O(t^{2d})$  with time  $t$ ) discretization required by the GP-UCB family of algorithms. Another factor contributing to the reduced complexity is the relaxed search criterion that aims to determine only the existence of threshold-exceeding points, in contrast to finding a global maximizer as in the GP-UCB family of algorithms. As a result, GP-ThreDS reduces the computational complexity from  $O(T^{2d+3})$  as required by GP-UCB family of algorithms to  $O(T^4)$ .

### 3.1.3 Related Work

There is a vast body of literature on numerical and theoretical analysis of Bayesian optimization algorithms. Following the work by Srinivas *et al.* [278] on GP-UCB, several extensions have been proposed based on combining GP with bandit techniques. Representative results include extensions to arbitrary compact metric spaces [72], contextual bandits [172, 308], parallel observations [92, 71], ordinal models [224], robust optimization [37], and multi-fidelity observations [155].

Chowdhury and Gopalan [68] proposed an improved version of GP-UCB with an improved confidence interval based on a self-normalized concentration inequality that was inspired by similar results derived by Abbasi-Yadikori *et al.* [1] for linear bandits. They also proved the same  $O(\gamma_T \sqrt{T})$  regret holds for GP-TS, a Bayesian Optimization algorithm based on Thompson sampling principle. Augmenting GP models with local polynomial estimators, Madison *et al.* [183] introduced LP-GP-UCB and established improved regret bounds for it under special cases (see, [183, Section 3.2]). However, for other cases, the regret guarantees for LP-GP-UCB remain in the same order as GP-UCB. More recently, Janz *et al.* [148] introduced  $\pi$ -GP-UCB, specific to Matérn family of kernels, that constructs a cover for the search space, as many hypercubes, and fits an independent GP to each cover element. This algorithm was proven to achieve sub-linear regret across all parameters of the Matérn family. Prior to this result, only SupKernelUCB [308] and RIPS [52] algorithms were known to achieve a regret of  $O(\sqrt{T\gamma_T})$ . However, their results are limited to discrete action spaces<sup>1</sup>. While

---

<sup>1</sup>In bandit terminology, the terms action and action spaces are used to refer to what are known as query point and domain in optimization literature. Throughout this thesis, we will use these terms interchangeably.

this may be extendable to continuous spaces via a discretization argument as recently pointed out in [148, 48], the required discretization needs to grow polynomially in  $T$ , making it computationally expensive. In fact, the query point selection strategy even in GP-UCB and its variants (that optimize over continuous action spaces) involves optimizing the UCB over the entire domain through an exhaustive search over a grid of  $O(t^{2d})$  points at time instant  $t$ . Moreover, it has been noted that SupKernelUCB performs poorly in practice [148, 49, 48]. As a result, existing algorithms are either known to achieve sub-optimal regret bounds or known to be computationally intensive. GP-ThreDS, on the other hand, is a computationally efficient algorithm that achieves tight regret bounds with good empirical performance (see Section 3.5). Following our work, Li and Scarlett [193] proposed a new algorithm called Batched Pure Exploration which also achieves optimal regret guarantees. Their algorithm, while simpler to implement than GP-ThreDS, also involves maximization of posterior variance over the entire domain, resulting in a high computational cost.

There is a growing body of work in the literature addressing the high cost associated with computing the posterior distribution (the first computational bottleneck as discussed in Section 3.1). Such approaches usually involve approximating the GP posterior by using techniques such as adaptive matrix sketching [49], sparse variational inference [291, 135, 303, 144], random Fourier features [227], linearization [174] and additivity [157]. These approaches are orthogonal to the approach adopted in this work. In particular, one can combine such sparse approximation techniques with GP-ThreDS to further reduce the computational complexity.

As pointed out earlier in Section 3.1, the dominating source of the compu-

tational cost is in finding the maximizer of the UCB score. This issue has not received much attention except in a couple of recent studies. Mutný et al. [214] considered a problem where the kernel can be approximated with Quadratic Fourier Features. This additional assumption results in a linear model where the UCB proxy can be optimized using an efficient global optimizer. However, this assumption practically limits the GP model to squared exponential kernels. In contrast, the computationally efficient approach proposed in this work is generally applicable. In [267], the authors proposed an adaptive discretization approach similar to that of [45, 326]. The key idea is to replace the uniform discretization in GP-UCB with a non-uniform discretization that adapts to the observed function values so that regions with higher function values enjoy a finer discretization. Nevertheless, the discretization is still carried over the entire function domain throughout the learning process, and a global maximization of the UCB score needs to be carried out at each time instant over a *linearly* growing set of discrete points. The proposed algorithm, GP-ThreDS, however, continuously shrinks the function domain and evaluates the UCB score always on a *bounded* set of discrete points. The global maximization objective is also relaxed to determining the existence of threshold-exceeding points.

Using a tree structure to represent successive partitions of the search domain is a classical approach and has seen its use in the bandit literature [45, 213, 166]. Such methods are characterized by growing the tree at nodes with high UCB without pruning the nodes with low values of UCB. This is fundamentally different from the domain shrinking approach of GP-ThreDS. A different tree-based method was considered in [317] where the tree structure is dynamic and may not be computationally efficient. Furthermore, they did not provide any theoretical results.

In contrast to our agnostic regularity assumption on  $f$  (being fixed and belonging to an RKHS), a Bayesian setting was also considered in [278], where  $f$  is assumed to be a sample from a GP. The regret bounds were then provided in high probability with respect to both noise and the randomness in  $f$ . For GP-UCB algorithm, Srinivas *et al.* [278] proved a tighter  $O(\sqrt{T\gamma_T})$  bound under the Bayesian setting. Under the Bayesian setting, [156] built on ideas from [243, 244] to show that GP-TS achieves the same order of regret as GP-UCB. A Bayesian optimization algorithm for max-value entropy search was shown to enjoy the same regret order as GP-UCB and GP-TS in [324]. An  $\Omega(\sqrt{T})$  lower bound on regret was proven under the Bayesian setting [256, 268]. The Bayesian setting is fundamentally different from the frequentist setting considered in our work.

Several works consider a noise-free setting which results in tighter regret bounds. In particular, Bull [46] studied the noise-free Bayesian optimization for Matérn family of kernels with the objective of minimizing simple regret . The simple regret problem can be addressed using pure exploration algorithms such as epsilon greedy. Under a Bayesian and noise free setting the regret bounds can further improved to exponential rates [88, 163, 119].

Practitioners' approach to Bayesian optimization generally consists of selecting the observation points based on optimizing the so called acquisition functions (such as UCB in GP-UCB and Thompson sample in GP-TS). Other notable acquisition functions are GP-EI (that stands for expected improvement) and GP-PI (that stands for probability of improvement) [137], which are shown to enjoy the same regret guarantees as GP-UCB [325, 217, 322]. When implementing Bayesian optimization algorithms which are based on an acquisition function (GP-UCB, TS, PI and EI), a practical idea is to use an off-the-shelf optimizer to

solve the optimization of the acquisition function at each iteration. This method although can lead to significant gains in computational complexity, invalidates the existing regret bounds. Our focus in this work has been to introduce a practical algorithm with provable regret guarantees.

## 3.2 Problem Statement

### 3.2.1 Problem Formulation

We consider the problem of optimizing a fixed and unknown function  $f : \mathcal{X} \rightarrow \mathbb{R}$ , where  $\mathcal{X} \subset \mathbb{R}^d$  is a convex and compact domain. The learning objective is to approach a maximizer,  $x^* \in \arg \max_{x \in \mathcal{X}} f(x)$ , of the function through a sequence of query points  $\{x_t\}_{t=1}^T$  chosen sequentially in time. Specifically, a sequential optimization algorithm chooses a point  $x_t \in \mathcal{X}$  at each time instant  $t = 1, 2, \dots$ , and observes  $y_t = f(x_t) + \epsilon_t$ , a noisy function value at that point. The noise sequence  $\{\epsilon_t\}_{t=1}^\infty$  is assumed to be i.i.d. over  $t$  and  $R$ -sub-Gaussian for a fixed constant  $R \geq 0$ , i.e.,  $\mathbb{E}[e^{\zeta \epsilon_t}] \leq \exp(\zeta^2 R^2 / 2)$  for all  $\zeta \in \mathbb{R}$  and  $t \in \mathbb{N}$ . The learning efficiency is measured by cumulative regret given by

$$R(T) = \sum_{t=1}^T [f(x^*) - f(x_t)]. \quad (3.1)$$

We assume a regularity condition on the objective function  $f$  that is commonly adopted under kernelized learning models. Specifically, we assume that  $f$  lives in a Reproducing Kernel Hilbert Space (RKHS) associated with a positive definite kernel  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ . The RKHS, denoted by  $H_k$ , is a Hilbert space associated with a positive definite kernel  $k(\cdot, \cdot)$  and is fully specified by

the kernel and vice versa. It is endowed with an inner product  $\langle \cdot, \cdot \rangle_k$  that obeys the reproducing property, i.e.,  $g(x) = \langle g, k(x, \cdot) \rangle_k$  for all  $g \in H_k$ . The inner product also induces a norm  $\|g\|_k = \sqrt{\langle g, g \rangle_k}$ . This norm is a measure of the smoothness of the function  $f$  with respect to the kernel  $k$  and is finite if and only if  $f \in H_k$ . The RKHS norm of  $f$  is assumed to be bounded by a known constant  $B$ , that is,  $\|f\|_k \leq B$ . We further assume that  $f$  is  $\alpha$ -Hölder continuous, that is,  $|f(x) - f(x')| \leq L\|x - x'\|^\alpha$  for all  $x, x' \in \mathcal{X}$  for some  $\alpha \in (0, 1]$  and  $L > 0$ . This is a mild assumption as this is a direct consequence of RKHS assumption for commonly used kernels as shown in [183]. We also assume the knowledge of an interval  $[a, b]$ , such that  $f(x^*) \in [a, b]$ . This is also a mild assumption as domain-specific knowledge often provides us with bounds. For example, a common application of black-box optimization is hyperparameter tuning in deep learning models. The unknown function represents the accuracy of the model for a given set of hyperparameters. Since  $f$  represents the accuracy of the model, we have  $f(x^*) \in [0, 1]$ . For simplicity of notation, we assume  $\mathcal{X} = [0, 1]^d$  and  $f(x^*) \in [0, 1]$ . It is straightforward to relax these assumptions to general compact domains and arbitrary bounded ranges  $[a, b]$ .

Our objective is to design a computationally efficient algorithm with a guarantee on regret performance as defined in (3.1). We provide high probability regret bounds that hold with probability at least  $1 - \delta_0$  for any given  $\delta_0 \in (0, 1)$ , a stronger performance guarantee than bounds on expected regret.

### 3.2.2 Preliminaries on Gaussian processes

Under the GP model, the unknown function  $f$  is treated hypothetically as a realization of a Gaussian process over  $\mathcal{X}$ . A Gaussian Process  $\{F(x)\}_{x \in \mathcal{X}}$  is fully specified by its mean function  $\mu(\cdot)$  and covariance function  $k(\cdot, \cdot)$ . All finite samples of the process are jointly Gaussian with mean  $\mathbb{E}[F(x_i)] = \mu(x_i)$  and covariance  $\mathbb{E}[(F(x_i) - \mu(x_i))(F(x_j) - \mu(x_j))] = k(x_i, x_j)$  for  $1 \leq i, j \leq n$  and  $n \in \mathbb{N}$  [228]. The noise  $\epsilon_t$  is also viewed as Gaussian.

The conjugate property of Gaussian processes with Gaussian noise allows for a closed-form expression of the posterior distribution. Consider a set of observations  $\mathcal{H}_t = \{\mathbf{x}_t, \mathbf{y}_t\}$  where  $\mathbf{x}_t = (x_1, x_2, \dots, x_t)^T$  and  $\mathbf{y}_t = (y_1, y_2, \dots, y_t)^T$ . Here  $y_s = f(x_s) + \epsilon_s$  where  $x_s \in \mathcal{X}$  and  $\epsilon_s$  are the zero-mean noise terms, i.i.d. over  $s$  for  $s \in \mathbb{N}$ . Conditioned on the history of observations  $\mathcal{H}_t$ , the posterior for  $f$  is also a Gaussian process with mean and covariance functions given as

$$\mu_t(x) = \mathbb{E}[F(x)|\mathcal{H}_t] = k_{\mathbf{x}_t, \mathbf{x}}^T (K_{\mathbf{x}_t, \mathbf{x}_t} + \lambda I)^{-1} \mathbf{y}_t \quad (3.2)$$

$$k_t(x, x') = \mathbb{E}[(F(x) - \mu_t(x))(F(x') - \mu_t(x'))|\mathcal{H}_t] = k(x, x') - k_{\mathbf{x}_t, \mathbf{x}}^T (K_{\mathbf{x}_t, \mathbf{x}_t} + \lambda I)^{-1} k_{\mathbf{x}_t, \mathbf{x}'} \quad (3.3)$$

In the above expressions,  $k_{\mathbf{x}_t, \mathbf{x}} = [k(x_1, x), \dots, k(x_t, x)]^T$ ,  $K_{\mathbf{x}_t, \mathbf{x}_t}$  is the  $t \times t$  covariance matrix  $[k(x_i, x_j)]_{i,j=1}^t$ ,  $I$  is the  $t \times t$  identity matrix and  $\lambda$  is the variance of the Gaussian model assumed for the noise terms.

Gaussian processes are powerful non-parametric Bayesian models for functions in RKHSs [154]. In particular, the mean function of the GP regression (eqn. (3.2)) lies in the RKHS with kernel  $k(\cdot, \cdot)$  with high probability. We emphasize that the GP model of  $f$  and the Gaussian noise assumption are internal to the learning algorithm. The underlying objective function  $f$  is an arbitrary

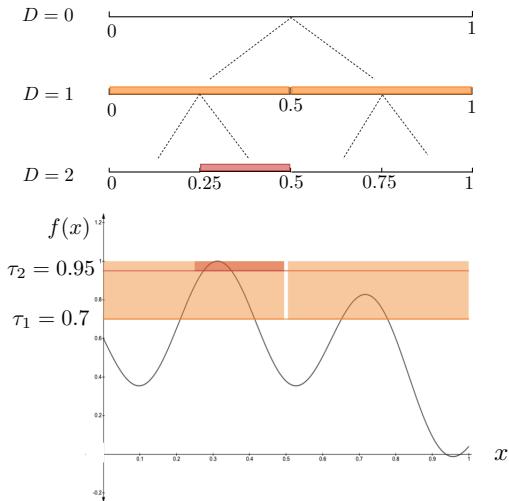


Figure 3.1: Thresholded domain shrinking.

deterministic function in an RKHS, and the noise obeys an arbitrary  $R$ -sub-Gaussian distribution.

### 3.3 The GP-ThreDS Algorithm

In Section 3.3.1, we present the basic domain-shrinking structure of GP-ThreDS that continuously prunes sub-performing regions of  $\mathcal{X}$  and zooms into increasingly smaller high-performing regions of  $\mathcal{X}$ . In Section 3.3.2, we present the method for identifying high-performing regions of  $\mathcal{X}$ .

#### 3.3.1 Thresholded domain shrinking

GP-ThreDS operates in epochs. Each epoch completes one cycle of pruning, refining, and threshold updating as detailed below. **(i) Pruning:** removing sub-performing regions of  $\mathcal{X}$  from future consideration; **(ii) Refining:** splitting high-

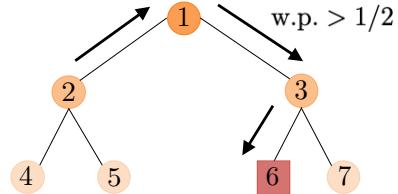


Figure 3.2: An illustration of the random-walk based search. (Node 6 is the single high-performing leaf node. If the random walk is currently at node 2, the correct direction is along the shortest path to node 6: via node 1 and then node 3.)

performing regions of  $X$  into smaller regions for refined search (i.e., zooming in) in future epochs; **(iii) Threshold updating:** updating the threshold on function values that defines the criterion for high/sub-performance to be used in the next epoch. The pruning and refining conform to a binary-tree representation of  $X$  with nodes representing regions of  $X$  and edges the subset relation (i.e., region splitting). Throughout the paper, we use nodes and regions of  $X$  interchangeably.

We explain the details with an example. Consider a one-dimensional function over  $X = [0, 1]$  as shown in Fig. 3.1. Assume that it is known  $f(x^*) \in [0, 1.4]$ . The function threshold  $\tau_1$  defining the pruning criterion in the first epoch is set to the mid-point:  $\tau_1 = 0.7$ . In epoch 1, the domain  $X$  is represented by a tree of height 1 with the root representing the entire domain  $[0, 1]$  and the two leaf nodes representing the two sub-intervals  $[0, 0.5]$  and  $(0.5, 1]$  (see Fig. 3.1). In the pruning stage of this epoch, the algorithm determines, with a required confidence, whether each leaf node contains a point with function value exceeding  $\tau_1$ . Such threshold-exceeding leaf nodes are referred to as high-performing nodes. Otherwise, they are called sub-performing nodes and are pruned, along with their ancestors, from the tree. Suppose that in this example, both sub-intervals  $[0, 0.5]$  and  $(0.5, 1]$  are identified as high-performing (see Section 3.3.2 on identifying high-performing nodes). Consequently, no node is pruned, and the algorithm proceeds to the refining stage, where each sub-interval splits, and the tree grows to a height of 2 with four leaf nodes. The threshold is then updated to  $\tau_2 = 0.95$  (see below on threshold updating). The increased threshold reflects an adjustment toward a more aggressive pruning in the next epoch as suggested by the presence of (multiple) high-performing nodes in the current epoch.

In the second epoch, the pruning stage aims to identify high-performing (defined by  $\tau_2$ ) nodes among the four leaf nodes. Supposed that it is determined leaf node  $(0.25, 0.5]$  is the only high-performing node. Then the nodes  $[0, 0.25]$ ,  $(0.5, 0.75]$  and  $(0.75, 1]$  and all their ancestors are pruned. In the refining stage, the high-performing node  $(0.25, 0.5]$  splits into two. The threshold is updated to  $\tau_3$ . The algorithm then progresses into the third epoch, facing the same decision problem on the two leaf nodes (the two children of  $(0.25, 0.5]$ ) of the pruned tree and following the same pruning-refining-threshold updating cycle.

For a general  $d$ -dimensional problem, the basic structure is the same with three simple generalizations. First, the two children of any given node are formed by equally splitting the longest edge of the corresponding  $d$ -dimensional cuboid (ties broken arbitrarily). Second, in each epoch, the tree grows by  $d$  levels ( $d = 1$  in the above example) in the refining stage by following successive binary splitting  $d$  times. The last detail to specify is that if no leaf node is identified as high-performing in an epoch  $k$ , then the refining stage is bypassed, and the algorithm repeats the search on the same tree (no pruning or refining) with a decreased threshold  $\tau_{k+1}$  in the next epoch. The decreased threshold reflects a lowered estimate of  $f(x^*)$  based on the absence of high-performing nodes in the current epoch.

The thresholds  $\{\tau_k\}_{k \geq 1}$  are updated iteratively using a binary search to approach  $f(x^*)$ . For each epoch  $k$ , the algorithm maintains an interval  $[a_k, b_k]$  which is believed to contain  $f(x^*)$ . The threshold  $\tau_k$  is set to the mid-point of  $[a_k, b_k]$ . The initial interval  $[a_1, b_1]$  is set to the known range  $[a, b]$  of  $f(x^*)$ . At the end of epoch  $k$ , if no leaf node is identified as high-performing, we set

$a_{k+1} = a_k - (b_k - a_k)/2$  and  $b_{k+1} = b_k - (b_k - a_k)/2$ , which leads to a decreased threshold in the next epoch. Otherwise, we set  $a_{k+1} = \tau_k - c2^{-\alpha\rho_k/d+1}$  and  $b_{k+1} = b_k$ , where  $\rho_k$  is the height of the tree before the pruning stage of epoch  $k$ , and  $c \in (0, 1/2)$  is a hyperparameter (specified in Section 3.3.2).

We emphasize that while the proposed domain-shrinking approach conforms to an ever-growing tree, the algorithm can be implemented without storing the entire tree. The only information about the tree that needs to be maintained is the set  $\mathcal{D}_k$  of high-performing leaf nodes identified in the pruning stage of each epoch  $k$ . A pseudo code for GP-ThreDS is given in Algorithm 2 below. In the pseudo-code, `getHighPerformingNodes` is the routine identifying the high-performing nodes on a tree of depth  $d$  that is the routine described in Sec 3.3.2.  $\mathcal{L}_v$  denotes the set of high-performing nodes returned by `getHighPerformingNodes` corresponding to the node  $v$  in  $\mathcal{D}_k$ .

### 3.3.2 Identifying high-performing nodes

We now specify the local algorithm for identifying high-performing nodes in a given epoch  $k$ . Recall that  $\mathcal{D}_k$  denotes the set of high-performing nodes identified in epoch  $k$ . Each node in  $\mathcal{D}_k$  has grown  $d$  levels and produced  $2^d$  leaf nodes in the refining stage of epoch  $k$ . The objective of epoch  $k+1$  is to determine which of the  $2^d |\mathcal{D}_k|$  newly grown leaves are high-performing nodes defined by  $\tau_{k+1}$ .

In epoch  $k + 1$ , the only portion of the tree that is of interest is the  $|\mathcal{D}_k|$  subtrees, each of height  $d$  with a root in  $\mathcal{D}_k$ . Our approach is to treat these subtrees separately, one at a time. We can thus focus on one subtree to describe the algorithm for identifying which of the  $2^d$  leaves are high-performing. The terms

---

**Algorithm 2** GP-ThreDS

---

```
Input:  $\mathcal{D}_0 = \{\mathcal{X}\}$ ,  $[a_1, b_1] = [0, 1]$ ,  $\delta_0 \in (0, 1)$ 
// The refining stage in epoch 0 is completed and we
have a tree with  $2^d$  leaves with root at  $\mathcal{D}_0$ .
Set  $k \leftarrow 1$ ,  $\rho_1 \leftarrow d$ ,  $\tau_1 = (a_1 + b_1)/2$ 
repeat
    Set  $\mathcal{D}_k \leftarrow \emptyset$ 
    // Start Pruning Stage
    for  $v$  in  $\mathcal{D}_{k-1}$  do
        Set  $\mathcal{T}_v$  to be the tree of depth  $d$  rooted at the node  $v \in \mathcal{D}_{k-1}$ 
         $\mathcal{L}_v \leftarrow \text{getHighPerformingNodes}(\mathcal{T}_v, \tau_k, \delta_0/4T)$ 
         $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \mathcal{L}_v$ 
    end for
    if  $\mathcal{D}_k = \emptyset$  then
        // No refining
         $\mathcal{D}_k \leftarrow \mathcal{D}_{k-1}$ ,  $\rho_{k+1} \leftarrow \rho_k$ 
         $a_{k+1} \leftarrow a_k - \frac{(b_k - a_k)}{2}$  and  $b_{k+1} \leftarrow b_k - \frac{(b_k - a_k)}{2}$ 
    else
        // Carry refining stage by growing subtrees rooted at
         $v$  for all  $v \in \mathcal{D}_k$ 
         $a_{k+1} \leftarrow \tau_k - c2^{-\alpha\rho_k/d+1}$ ,  $b_{k+1} \leftarrow b_k$ ,  $\rho_{k+1} \leftarrow \rho_k + d$ 
    end if
    // Update the threshold
     $\tau_{k+1} = (a_{k+1} + b_{k+1})/2$ 
     $k \leftarrow k + 1$ 
until query budget is exhausted
```

---

root, node, and leaf all pertain to this subtree. We also omit the epoch index for simplicity.

### A random-walk based search for high-performing nodes

A straightforward approach to identifying the high-performing nodes is to test each of the  $2^d$  leaf nodes directly. This, however, results in a large number of samples at suboptimal points when the dimension  $d$  is high. Our approach is inspired by the RWT (Random Walk on a Tree) algorithm recently proposed as a robust and adaptive algorithm for stochastic convex optimization [304, 306,

253].

Assume first there is exactly one high-performing node among the  $2^d$  leaf nodes. The basic idea is to devise a biased random walk on the tree that initiates at the root and walks towards the high-performing node at the leaf level. As illustrated in Fig. 3.2 with  $d = 2$ , at a non-leaf node, the random walk can take one of three directions: towards the parent or one of the two children (the parent of the root is itself). The correct direction is to walk along the shortest path to the high-performing leaf node. With the subset relation encoded by the tree, this implies moving to the child containing a threshold-exceeding point or to the parent when neither child contains threshold-exceeding points. Hence, to guide the random walk, a local sequential test is carried out on the two children, one at a time, to determine, at a required confidence level, whether it is threshold-exceeding (see Section 3.3.2). The walk then moves to the first child identified as threshold-exceeding (if any) or to the parent otherwise. The confidence level of the local sequential test at each non-leaf node is only required to ensure the walk is correctly biased, i.e., the probability of walking in the correct direction is greater than 1/2.

On reaching a leaf node, the algorithm enters the *verification* stage to determine whether this node is the high-performing leaf node. If the decision is no, it moves back to the parent of this leaf node, and the random walk resumes. If yes, the algorithm exits (under the assumption of a single high-performing leaf node). This decision can be made by carrying out the same local sequential test that guides the random walk at non-leaf nodes. The only difference is in the required confidence level. Given that a false positive at a leaf cannot be corrected due to exiting while a false negative only resumes the random walk (hence re-

tractable in the future), the confidence level for a positive decision needs to be sufficiently high to ensure the overall regret performance, while a negative decision only needs to ensure the bias of the walk (as in the non-leaf nodes). A pseudo code for the random-walk based search strategy for the case of identifying a single high-performing node with confidence level  $\delta_{RW}$  in Alg. 3. As the names suggest, the function `root` returns the root node of the tree, `parent`, `leftChild` and `rightChild` return the parent node, the left child and the right child, respectively, of the node in the argument. `SequentialTest` is the sequential test routine described in Section 3.3.2. In addition to the node and threshold, it takes confidence parameter for the local test as the input.

When the number of high-performing leaf nodes is unknown and arbitrary in  $\{0, 1, 2, \dots, 2^d\}$ , multiple runs of the random walk are carried out to identify them one by one. In addition, a termination test on the root node is carried out before each run to determine whether there are still unidentified high-performing leaf nodes. A pseudo code is described in Alg. 4. In Alg. 4 a high-performing node is identified, it is not considered in future iterations, in order to avoid redetection. Specifically, while carrying out the sequential test on any node, during the  $(r + 1)^{\text{th}}$  iteration, on any ancestor of the  $r$  identified nodes, the points belonging to the identified high-performing nodes are not considered. This is reflected in updating  $\mathcal{T}$  to  $\mathcal{T} \setminus \text{retNode}$ . The parameter  $\delta_{RW}$  is set to  $\delta_0/4T$ .

We emphasize that the local test is carried out using only observations from the current visit to this node; observations from past visits are forgotten. This is to ensure the random-walk nature of the process for tight-analysis. A computational benefit is that the matrices being inverted to compute the posterior distribution are always small, improving the run-time efficiency of the algo-

---

**Algorithm 3** Random-walk based strategy for one high-performing node

---

**Input:** Binary tree  $\mathcal{T}$  of depth  $d$ , threshold  $\tau$ , confidence level  $\delta_{RW}$ .  
Set  $currNode \leftarrow \text{root}(\mathcal{T})$ ,  $terminate \leftarrow 0$

**while**  $terminate \neq 1$  **do**

**if**  $\text{depth}(currNode) == d$  **then**  
     $retLeaf \leftarrow \text{SequentialTest}(currNode, \tau, p)$   
    **if**  $retLeaf == 1$  **then**  
         $terminate \leftarrow 1$   
         $retNode \leftarrow currNode$   
    **else**  
         $currNode \leftarrow \text{parent}(currNode)$   
    **end if**  
**else**  
     $retLeft \leftarrow \text{SequentialTest}(\text{leftChild}(currNode), \tau, p)$   
    **if**  $retLeft == 1$  **then**  
         $currNode \leftarrow \text{leftChild}(currNode)$   
    **else**  
         $retRight \leftarrow \text{SequentialTest}(\text{rightChild}(currNode), \tau, p)$   
        **if**  $retRight == 1$  **then**  
             $currNode \leftarrow \text{rightChild}(currNode)$   
        **else**  
             $currNode \leftarrow \text{parent}(currNode)$   
        **end if**  
    **end if**  
**end if**  
**end while**  
**return**  $retNode$

---

rithm.

### The local sequential test

The last piece of the puzzle in GP-ThreDS is the local sequential test on a given node of a subtree. Given a node/region,  $D \subseteq \mathcal{X}$ , a threshold  $\tau$ , and a confidence parameter  $\eta \in (0, 1)$ , the local sequential test needs to determine, with a  $1 - \eta$  confidence level, whether  $D$  contains a point with function value exceeding  $\tau$ .

The test first builds a discretization of the region  $D$ , denoted by the set  $D_g =$

---

**Algorithm 4** Random-walk based strategy for multiple high-performing node

---

**Input:** Binary tree  $\mathcal{T}$  of depth  $d$ , threshold  $\tau$ , confidence level  $\delta_{RW}$ .  
Set currNode  $\leftarrow \text{root}(\mathcal{T})$ , terminate  $\leftarrow 0$ ,  $r \leftarrow 1$ , HPNodes  $\leftarrow \emptyset$ , retNode = NULL

**while** terminate  $\neq 1$  **do**

**if** currNode == root( $\mathcal{T}$ ) **then**

retRoot  $\leftarrow \text{SequentialTest}(\text{currNode}, \tau, p)$

**if** retRoot == -1 **then**

terminate  $\leftarrow 1$

**end if**

**else**

**if** depth(currNode) ==  $d$  **then**

retLeaf  $\leftarrow \text{SequentialTest}(\text{currNode}, \tau, p)$

**if** retLeaf == 1 **then**

retNode  $\leftarrow \text{currNode}$

**else**

currNode  $\leftarrow \text{parent}(\text{currNode})$

**end if**

**else**

retLeft  $\leftarrow \text{SequentialTest}(\text{leftChild}(\text{currNode}), \tau, p)$

**if** retLeft == 1 **then**

currNode  $\leftarrow \text{leftChild}(\text{currNode})$

**else**

retRight  $\leftarrow \text{SequentialTest}(\text{rightChild}(\text{currNode}), \tau, p)$

**if** retRight == 1 **then**

currNode  $\leftarrow \text{rightChild}(\text{currNode})$

**else**

currNode  $\leftarrow \text{parent}(\text{currNode})$

**end if**

**end if**

**end if**

**if** retNode != NULL **then**

HPNodes  $\leftarrow \text{HPNodes} \cup \text{retNode}$

$\mathcal{T} \leftarrow \mathcal{T} \setminus \text{retNode}$

$r \leftarrow r + 1$

retNode = NULL

**end if**

**end if**

**end while**

**return** HPNodes

---

$\{x_i\}_{i=1}^{|D_g|}$ . The set of points in  $D_g$  are chosen to ensure that  $\sup_{x \in D} \inf_{y \in D_g} \|x - y\| \leq \Delta$ . A simple way to construct such a discretization is to use uniform grids parallel to the axes with a resolution small enough to satisfy the above constraint. The parameter  $\Delta$  in epoch  $k$  is set to  $\Delta_k = (c/L)^{1/\alpha} 2^{-\rho_k/d}$  and is used to control the approximation of the function values in  $D$ . Recall that  $L$  is the Hölder continuity constant while  $c \in (0, 1/2)$  is a hyperparameter. The local test sequentially queries points in the set  $D_g$  to locally estimate  $f$ .

To determine whether there exists a point  $x \in D$  with  $f(x) \geq \tau$ , the test builds a pair of Upper and Lower Confidence Bounds using sequentially drawn samples and compares each of them to prescribed values. If the UCB goes below  $\tau - L\Delta^\alpha$ , indicating that the node is unlikely to contain a  $\tau$ -exceeding point, the test terminates and outputs a negative outcome. On the other hand, if LCB exceeds  $\tau$ , then this is a  $\tau$ -exceeding point with the required confidence level. The test terminates and outputs a positive outcome. If both the UCB and LCB are within their prescribed “uncertainty” range, the test draws one more sample and repeats the process. A cap is imposed on the total number of samples. Specifically, the test terminates and outputs a positive outcome when the total number of samples exceeds  $\bar{S}(p, L\Delta^\alpha)$ . A description of the test for  $s \geq 1$  after being initialized with a point  $x_1 \in D_g$  is given in Fig. 3.3. We would like to emphasize that the posterior mean and variance  $\mu_{s-1}$  and  $\sigma_{s-1}^2$  considered in the description below are constructed only from the samples collected during that particular visit to the current node.

The parameter  $\beta_s(\nu) := B + R \sqrt{2(\gamma_{s-1} + 1 + \log(1/\nu))}$  for  $\nu \in (0, 1)$ .  $\gamma_t$  is the maximum information gain at time  $t$ , defined as  $\gamma_t := \max_{A \subset \mathcal{X}: |A|=t} I(y_A; f_A)$ . Here,  $I(y_A; f_A)$  denotes the mutual information between  $f_A = [f(x)]_{x \in A}$  and  $y_A = f_A + \epsilon_A$ .

- If  $\max_{x \in D_g} \mu_{s-1}(x) - \beta_s(\eta)\sigma_{s-1}(x) \geq \tau$ , terminate and output +1.
  - If  $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\eta)\sigma_{s-1}(x) \leq \tau - L\Delta^\alpha$ , terminate and output -1.
  - Otherwise, query  $x_s = \arg \max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\frac{\delta_0}{4T})\sigma_{s-1}(x)$
  - Observe  $y_s = f(x_s) + \epsilon_s$  and use (3.2) and (3.3) to obtain  $\mu_s$  and  $\sigma_s$ . Increment  $s$  by 1.
  - Repeat until  $s < \bar{S}(\eta, L\Delta^\alpha)$ .
- If  $s = \bar{S}(\eta, L\Delta^\alpha)$ , terminate and output +1.

Figure 3.3: The local sequential test for the decision problem of finding a  $\tau$ -exceeding point.

Bounds on  $\gamma_t$  for several common kernels are known [279, 301] and are sublinear functions of  $t$ .

The cap  $\bar{S}(\eta, L\Delta^\alpha)$  on the maximum number of samples is given by

$$\bar{S}(\eta, L\Delta^\alpha) = \min \left\{ t \in \mathbb{N} : \frac{2(1+2\lambda)\beta_t(\eta)|D_g|^{\frac{1}{2}}}{(L\Delta^\alpha)\sqrt{t}} \leq 1 \right\} + 1. \quad (3.4)$$

The cap on the total number of samples prevents the algorithm from wasting too many queries on suboptimal nodes. Without such a cap, the expected number of queries issued by the local test is inversely proportional to  $|f(x_{D_g}^*) - \tau|$ , where  $x_{D_g}^* = \arg \max_{x \in D_g} f(x)$ . Consequently, small values of  $|f(x_{D_g}^*) - \tau|$  would lead to a large number of queries at highly suboptimal points when  $f(x_{D_g}^*)$  is far from  $f(x^*)$ . The cap on the number of samples thus helps control the growth of regret at the cost of a potential increase in the approximation error. It also reduces the cost in computing the posterior distribution by limiting the number of queries at a node.

Note that when the sequential test reaches the maximum allowable samples and exits with an outcome of +1, it is possible that  $f(x_{D_g}^*) < \tau$  (i.e., no  $\tau$ -exceeding points in  $D_g$ ). Thus,  $\tau$  may not be a lower bound for the updated belief of  $f(x^*)$ , as one would expect in the case of an output of +1 from the sequential test.

However, using Lemma 3.4.3, we can obtain a high probability lower bound on  $\tau - f(x_{D_g}^*)$ . This additional error term is taken into account while updating the threshold as described in Section 3.3.1. The hyperparameter  $c$  trades off this error with the size of the discretization.

The sequential test can be easily modified to offer asymmetric confidence levels for declaring positive and negative outcomes (as required for in the verification stage of the RWT search) by changing the confidence parameter in  $\beta_s$ . Please refer to Appendix B.1.2 for additional details.

We point out that the construction of the UCB is based on the UCB score employed in IGP-UCB [68]. It is straightforward to replace it with other types of UCB scores. The basic thresholded domain shrinking structure of the proposed algorithm is independent of the specific UCB scores, hence generally applicable as a method for improving the computational efficiency and regret performance of GP-UCB family of algorithms.

### 3.4 Performance Analysis

In this section, we analyze the regret and computational complexity of GP-ThreDS. Throughout the section,  $D \subseteq \mathcal{X}$  denotes a node visited by GP-ThreDS,  $D_g$  denotes its associated discretization, constructed as described in Section 3.3.2, and  $x_{D_g}^* = \arg \max_{x \in D_g} f(x)$ .

### 3.4.1 Regret Analysis

The following theorem establishes the regret order of GP-ThreDS.

**Theorem 3.4.1.** *Consider the GP-ThreDS algorithm as described in Section 3.3. Then, for any  $\delta_0 \in (0, 1)$ , with probability at least  $1 - \delta_0$ , the regret incurred by the algorithm is given as*

$$R(T) = O(\sqrt{T\gamma_T} \log T (\log T + \sqrt{\log T \log(1/\delta_0)})).$$

We provide here a sketch of the proof. The regret incurred by GP-ThreDS is analysed by decomposing it into two terms: the regret in the first  $k_0$  epochs referred to as  $R_1$ , and the regret after the completion of the first  $k_0$  epochs referred to as  $R_2$ , where  $k_0 = \max\{k : \rho_k \leq \frac{d}{2\alpha} \log T\}$ . To bound  $R_1$ , we first bound the regret incurred at any node visited during the first  $k_0$  epochs using the following decomposition of the instantaneous regret:

$$f(x^*) - f(x_t) = [f(x^*) - \tau_k + L\Delta_k^\alpha] + [\tau_k - f(x_{D_g}^*) - L\Delta_k^\alpha] + [f(x_{D_g}^*) - f(x_t)].$$

In the above decomposition,  $k$  denotes the epoch index during which the node is visited. Each of these three terms are then bounded separately. The third term in the expression is bounded using a similar approach to the analysis of IGP-UCB [68] (notice that  $x_t$  is the maximizer of the UCB score), that is, to bound it by the cumulative standard deviation ( $\sum_{s=1}^t \sigma_{s-1}(x_s)$ ).

**Lemma 3.4.2.** *For any set of sampling points  $\{x_1, x_2, \dots, x_t\}$  chosen from  $D_g$  (under any choice of algorithm), the following relation holds:  $\sum_{s=1}^t \sigma_{s-1}(x_s) \leq (1 + 2\lambda) \sqrt{|D_g|t}$ , where  $\sigma_s(x)$  is defined in (3.3).*

Since GP-ThreDS ensures a constant-sized discretization at all times (See Lemma 3.4.7), the above lemma implies that the sum of posterior standard devi-

ations is  $O(\sqrt{t})$  resulting in a tight bound corresponding to the third term (that is an  $O(\sqrt{\gamma_t})$  tighter than the bound for IGP-UCB which optimizes the UCB score over the entire domain). The first two terms are bounded using the following lemma with an appropriate choice of  $\Delta_f$ .

**Lemma 3.4.3.** *If the local test is terminated by the termination condition at instant  $\bar{S}(\delta_2, \Delta_f)$  as defined in Eqn. (3.4), then with probability at least  $1 - \delta_2$ , we have  $\tau - L\Delta^\alpha - \Delta_f \leq f(x_{D_g}^*) \leq \tau + \Delta_f$ .*

The final bound on  $R_1$  is obtained by a combination of the upper bound on regret on each node and the bound on the total number of nodes visited by GP-ThreDS, captured in the following lemma.

**Lemma 3.4.4.** *Consider the random walk based routine described in Section 3.3.2 with a local confidence parameter  $p \in (0, 1/2)$ . Then with probability at least  $1 - \delta_1$ , one iteration of RWT visits less than  $\frac{\log(d/\delta_1)}{2(p - 1/2)^2}$  nodes before termination.*

To bound  $R_2$ , we bound the difference in function values using the Hölder continuity of the function along with the upper bound on the diameter of the nodes after  $k_0$  epochs. Adding the bounds on  $R_1$  and  $R_2$ , we arrive at the theorem. The detailed proofs are provided in Appendix B.2. We would like to point out that the regret analysis depends on the choice of the UCB score. While we have used the UCB score of IGP-UCB, this analysis is straightforward to extend to other UCB scores.

*Remark 3.4.5.* We note that our assumptions are consistent with those used in proving the lower bounds. In particular, the lower bounds are proven for the Matérn family of kernels including the SE kernel in [257]. Proposition 1 in [183] proves the Hölder continuity of this family of kernels. Thus, our assumption on

Hölder continuity is consistent with the lower bound. In addition, the proof of lower bound considers a class of functions whose RKHS norm is upper bounded by a known constant [257, Sec 1.1]. This upper bound translates to an upper bound on the absolute value of  $f$ , which is consistent with our assumption on having a finite range for  $f$ .

### 3.4.2 Computational Complexity

The following theorem bounds the worst-case overall computational complexity of GP-ThreDS.

**Theorem 3.4.6.** *The worst-case overall computational complexity of GP-ThreDS is  $O(T^4)$ , where  $T$  is the time horizon.*

The proof of theorem follows from the following lemma.

**Lemma 3.4.7.** *The number of points in the discretization,  $|D_g|$ , for any node  $D$ , is upper bounded by a constant, independent of time. i.e.,  $|D_g| = O(1)$ ,  $\forall t \leq T$ .*

From the lemma, we can conclude that the number of UCB score evaluations in GP-ThreDS is constant at all times  $t$ , hence matrix inversion becomes the dominant source of computational complexity. Since no more than  $t$  samples are used to compute the posterior distribution at time  $t$ , the worst-case cost associated with matrix inversion step is  $O(t^3)$  and consequently the worst-case computational complexity of GP-ThreDS is  $O(T^4)$  leading to computational savings of  $O(T^{2d-1})$  over GP-UCB family of algorithms. Lemma 3.4.7 is proven by showing that the rate of domain size shrinking matches the rate of granularity

of the discretization across epochs. Thus, the size of discretization does not need to increase with time. Please refer to Appendix B.2 for a detailed proof.

While the discretization does not grow with  $t$ , it is exponential in  $d$ . Since non-convex optimization is NP-Hard, such an exponential dependence on  $d$  is inevitable for maintaining the optimal learning efficiency. In this work, we focus on reducing the computational complexity with respect to the time horizon  $T$ . The proposed domain shrinking technique can be used in conjunction with dimension reduction techniques (e.g., [323]) to achieve efficiency in both  $T$  and  $d$  (although at the price of invalidating the regret bounds).

### 3.5 Empirical Studies

In this section, we compare the performance of GP-ThreDS with several commonly used Bayesian optimization algorithms: IGP-UCB [68], Adaptive Discretization (AD) [267], Expected Improvement (EI) [322] and Probability of Improvement (PI) [325]. For the local test of GP-ThreDS we use the exact same UCB score as the one in IGP-UCB. We compare the performance of these algorithms on commonly used benchmark functions as well as on a real world task of hyperparameter tuning in a convolutional neural network.

#### 3.5.1 Experiments on Benchmark Functions

We used two standard benchmark functions, Branin and Rosenbrock in our experiments. The analytical expression for these functions is given below [20, 225]

- *Branin:*  $f(x, y) = -\frac{1}{51.95} \left( \left( v - \frac{5.1u^2}{4\pi^2} + \frac{5u}{\pi} - 6 \right)^2 + \left( 10 - \frac{10}{8\pi} \right) \cos(u) - 44.81 \right)$ , where  $u = 15x - 5$  and  $v = 15y$ .
- *Rosenbrock:*  $f(x, y) = 10 - 100(v - u)^2 - (1 - u)^2$ , where  $u = 0.3x + 0.8$  and  $v = 0.3y + 0.8$ .

. For the experiments, we use the SE kernel with lengthscale of  $l = 0.2$  on domain  $[0, 1]^2$  to model the functions. We use a Gaussian noise with variance of 0.01. The parameters  $\lambda$  in the GP model and  $R$  in  $\beta_t$  are also set to 0.01. The value of  $\delta_0$  is set to  $10^{-3}$ . To limit the computational cost in the standard implementation of IGP-UCB, we consider a maximum of 6400 points in the grid. The implementation details of the various algorithms are provided below.

1. IGP-UCB: The algorithm is implemented exactly as outlined in [68] with  $B$  (in scaling parameter  $\beta_t$ ) set to 0.5 and 2 for Branin and Rosenbrock, respectively. The parameters  $R$  and  $\delta_0$  are set to  $10^{-2}$  and  $10^{-3}$  in both experiments.  $\gamma_t$  was set to  $\log t$ . The size of discretization is increased over time, starting from 400 points at the beginning and capped at 6400 points.
2. Adapative Discretization (AD): The algorithm and its parameters are implemented exactly as described in [267].
3. Expected Improvement(EI)/Probability of Improvement (PI): Similar to IGP-UCB, EI and PI select the observation points based on maximizing an index often referred to as an acquisition function. The acquisition function of EI is  $(\mu(x) - f^* - \varepsilon)\Phi(z) + \sigma(x)\phi(z)$ , where  $z = \frac{\mu(x) - f^* - \varepsilon}{\sigma(x)}$ . The acquisition function of PI is  $\Phi(z)$ , where  $z = \frac{\mu(x) - f^* - \xi}{\sigma(x)}$ . Here,  $\Phi(\cdot)$  and  $\phi(\cdot)$  denote the CDF and PDF of a standard normal random variable.  $f^*$  is set to be

the maximum value of  $\mu(x)$  among the current observation points. The parameters  $\varepsilon$  and  $\xi$  are used to balance the exploration-exploitation trade-off. We follow [137] that showed the best choice of these parameters are small non-zero values. In particular,  $\varepsilon$  and  $\xi$  are both set to 0.01.

4. GP-ThreDS: A pseudo-code for the implementation is given in Alg. 22. The parameter  $\beta_t$  is set exactly in the same way as in IGP-UCB. The initial interval  $[a, b]$  is set to  $[0.5, 1.2]$  for Branin and  $[3, 12]$  for Rosenbrock. The parameter  $c$  is set to 0.2 for both functions.

Figures 3.4a, 3.4b and 3.4c, show the per-sample average regret, in log scale, measured at every 0.1 seconds, wall clock time. Specifically, within a given time (the X-axis of the figures), different algorithms process different number of samples determined by their computational complexity. The average per sample regret is then shown against the time taken. The plots are the average performance over 10 Monte Carlo runs. As expected from theoretical results, GP-ThreDS achieves the best performance especially as time grows. Figure 3.4d directly compares the computation time of all algorithms for processing 1000 samples, averaged over 10 Monte Carlo runs. GP-ThreDS enjoys a much smaller computation cost in terms of time taken (in seconds).

### 3.5.2 Hyperparameter Tuning of a convolutional neural network

In this section, we compare the performance of different Bayesian optimization algorithms for the task hyperparameter tuning of a convolutional neural network (CNN) on an image classification task. We have considered the task of

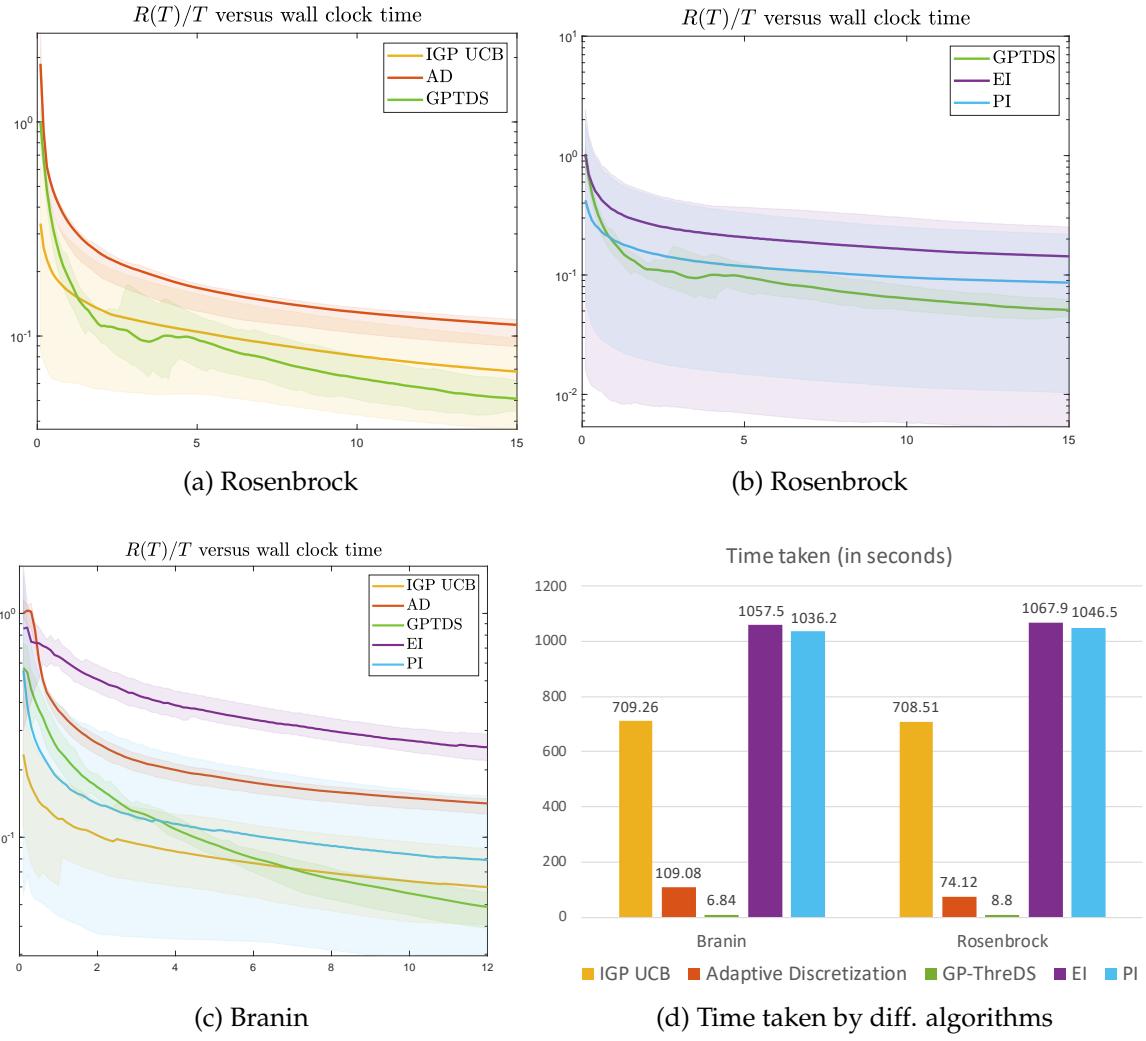


Figure 3.4: (a)-(c) Average cumulative regret against wall clock time for different algorithms on benchmark functions ((a)-(b) Rosenbrock, (c) Branin). (d) Computation time (in seconds) for 1000 samples for different algorithms.

digit classification on the MNIST dataset [180]. For the experiments, we have considered a smaller training dataset that contains only 12000 images instead of 50000 images in the original dataset. This smaller training dataset is created by randomly sampling 1200 images corresponding to each digit, making a total of 12000 images. This dataset is split to training and validation sets of size 10000 and 2000, respectively. The split is done in a way that each label has equal numbers of images in the training and the validation set. We used the same test set of 10000 images as in the original MNIST data set.

We consider a simple CNN with 2 convolutional layers followed by two fully connected feedforward layers. We use the ReLU activation function and a max pooling layer with stride 2 after each convolutional layer. The performance of the algorithms is evaluated on the task of tuning the following five hyperparameters of this CNN.

- Batch size: We considered 8 possible values of the batch sizes given by  $\{2^3, 2^4, \dots, 2^{10}\}$ .
- Kernel size of the first convolutional layer with possible values in  $\{3, 5, 7, 9\}$ .
- Kernel size of the second convolutional layer with possible values in  $\{3, 5, 7, 9\}$ .
- Number of (hidden) nodes in the first feedforward layer: The possible values for this hyperparameter belonged to  $\{10, 11, 12, \dots, 38, 39, 40\}$ .
- Initial learning rate: We used stochastic gradient descent with momentum to optimize the loss function. This parameter defined the initial learning rate for the optimizer and it took values in  $\{10^{-6}, 10^{-5}, \dots, 10^{-1}\}$ .

In the implementation, all these parameters were mapped to  $[0, 1]$  with distinct intervals corresponding to each discrete value. The kernel sizes and the number of hidden nodes were mapped linearly to the interval while the other two parameters were mapped on a log scale, that is,  $\log_2(\text{batch-size})$  and  $\log_{10}(\text{learning-rate})$  were mapped uniformly to the interval  $[0, 1]$ .

For this task, the underlying function was modelled using a Matérn kernel with smoothness parameter 2.5 and lengthscale  $l = 0.2$ . For this kernel  $\gamma_t$  was set to  $\sqrt{t}$  and the noise variance was set to 0.0001. The implementation of all

the algorithms is similar to the description in Section 3.5.1. For the exploration parameter  $\beta_t$ ,  $B$  and  $R$  are set to 0.5 and  $10^{-4}$ , respectively, for both IGP-UCB and GP-ThreDS. The confidence parameter  $\delta_0$  is set to  $2.5 \times 10^{-2}$  for IGP-UCB and  $2 \times 10^{-2}$  for GP-ThreDS. The slightly higher confidence for GP-ThreDS is chosen because it runs for a longer horizon to achieve the same compute time. In GP-ThreDS, the interval  $[a, b]$  is set to  $[0.3, 1.4]$  and  $c$  is set to 0.1.

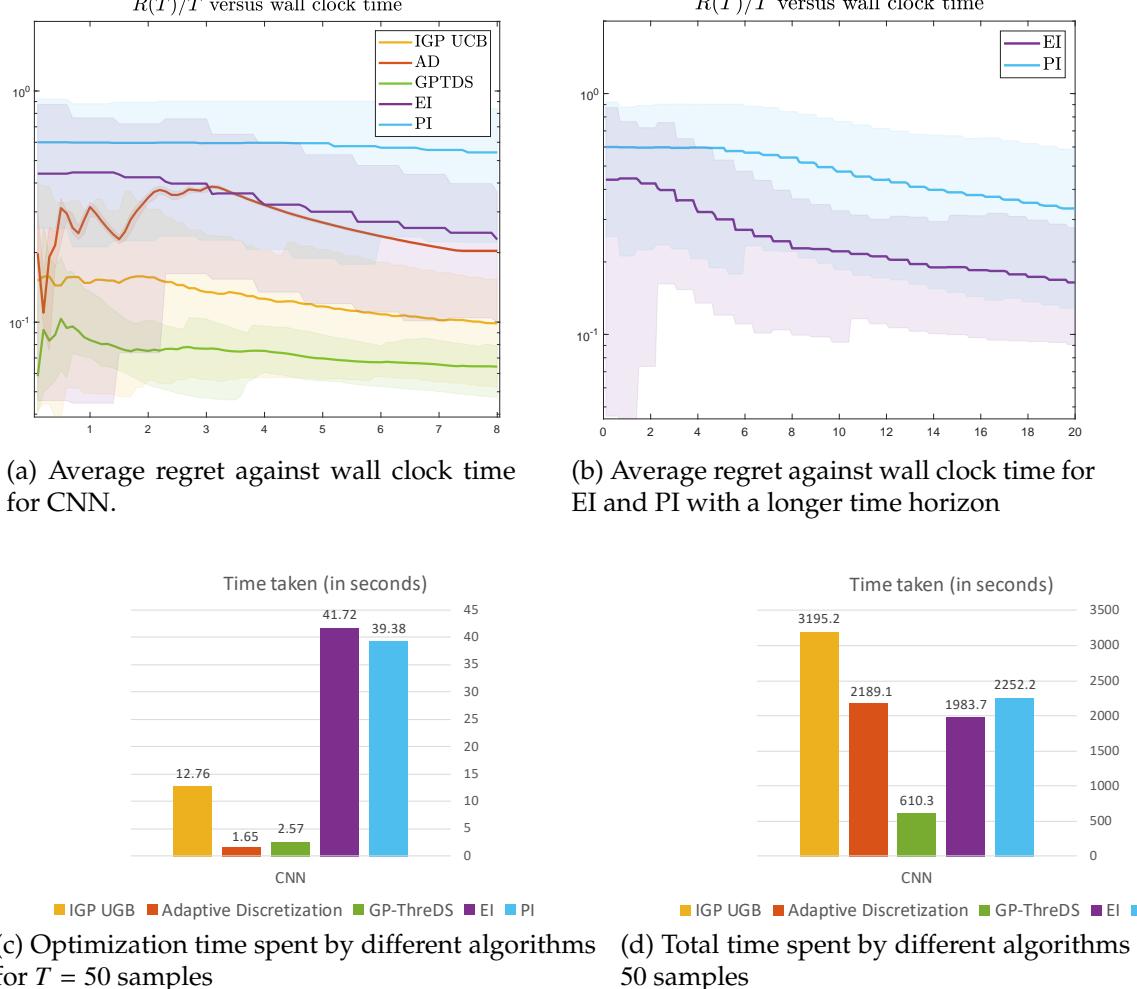


Figure 3.5: Performance of different algorithms for tuning the hyperparameters of a CNN for image classification.

The results for average regret against wall clock time, averaged over 10 Monte Carlo runs, are shown in Fig. 3.5a. As it can be seen from the figure,

GP-ThreDS offers a better performance compared to all other algorithms. It is important to point out that the time on  $X$ -axis does not include the time spent to train the CNN, it only includes the time spent on Bayesian optimization algorithms assuming the values of the objective function  $f$  (here the performance of CNN) are accessible in zero time. To be clear we refer to this time as *Optimization time*. The optimization time spent by different algorithms for  $T = 50$  is shown in Fig. 3.5c. We also report the corresponding *total time* contrasting the optimization time that also includes the time spent to train the CNN, in Fig. 3.5d. It can be seen that the total time for GP-ThreDS is significantly lower than all other algorithms. Its optimization time however is comparable to AD, while significantly lower than IGP-UCB, EI, PI. It seems that, due to its exploration scheme, AD selects hyperparameters which lead to longer training times. PI and EI in comparison seem to take longer to converge as shown in Fig. 3.5b.

## CHAPTER 4

### NEURAL NETWORK OPTIMIZATION

#### 4.1 Introduction

The stochastic contextual bandit has been extensively studied in the literature (see, [177, 178] and references therein). In this problem, at each discrete sequential step, a *context* is revealed by the environment. Then, a bandit agent chooses one of the  $K$  available actions. Choosing each action yields a random reward, the distribution of which is determined by the context. This is a variation of the classical bandit (or optimization setup) where the available set of actions (or subset of domain/query points) varies across time and depends on the *context*. Consequently, this “contextual version” of the problem offers an additional challenge in designing learning algorithms.

The problem was primarily studied under a linear setting where the expected reward of each arm is a linear function of the context [69, 192, 66]. It was later extended to kernel-based models [308], where the expected reward function associated with the context-action pairs belongs to the reproducing kernel Hilbert space (RKHS) determined by a known kernel. Recently, a variation of the problem has been proposed where the expected reward function associated with the context-action pairs is modeled using a neural net [346, 341, 120, 162]. The three contextual bandit settings mentioned above are increasingly more complex in the following order: *Linear* → *Kernel-based* → *Neural*.

Contextual bandits find application in recommender systems, information retrieval, healthcare and finance (see a survey on applications in [40]). The

problem can also be seen as a middle step from classic stochastic bandits [15] and (non-contextual) linear bandits [1] towards a reinforcement learning (RL) problem on a Markov decision process (MDP), where the contexts resemble the states of the MDP. One of the first formulations of linear contextual bandits was referred to as *associative RL* with linear value function [14]. Nonetheless, contextual bandit is different from RL in that the contexts are determined arbitrarily (adversarially) rather than following a stochastic Markovian process.

#### 4.1.1 Existing Results on Neural Contextual Bandits

With neural nets demonstrating a great representation power (much more general than linear models), there has been an increasing interest in modeling bandit and RL problems based on neural nets. This setting can be implemented using the typical neural net toolboxes. The neural contextual bandit has been considered in several works: Zhou *et al.* [346] and Zhang *et al.* [341], respectively, adopted the upper confidence bound (UCB) and Thompson sampling (TS) algorithms to the neural bandit setting. The algorithms are, respectively, referred to as NeuralUCB and NeuralTS. Gu *et al.* [120] considered a batch observation setting, where the reward function can be evaluated only on batches of data. Kassraie and Krause [162] provided analysis for a variation of NeuralUCB algorithm with diminishing instantaneous regret.

The analysis of neural bandits has been enabled through the theory of neural tangent (NT) kernel [147], which approximates an overparameterized neural net with the kernel corresponding to its infinite width, and the bounds on the approximation error (e.g., see, [13]). Zhou *et al.* [346], Zhang *et al.* [341] and Gu *et*

*al.* [120] proved  $\tilde{O}(\Gamma_k(T) \sqrt{T})$ <sup>1</sup> bounds on cumulative regret, where  $T$  is the number of steps and  $\Gamma_k(T)$  is a complexity term determined by the neural tangent kernel  $k$  associated with the particular neural network. Specifically,  $\Gamma_k(T)$  is the information gain corresponding to  $k$  for datasets of size  $T$ . For the ReLU neural nets on  $d$ -dimensional domains considered in [346, 341, 120],  $\Gamma_k(T) = \tilde{O}(T^{\frac{d-1}{d}})$  is the best upper bound known for the information gain [162, 300]. Inserting this bound on  $\Gamma_k(T)$ , the aforementioned regret bounds become trivial (superlinear) generally when  $d > 1$ , unless further restrictive assumptions are made on the contexts. Kassraie and Krause [162] addressed this issue by considering the *Sup* variant of NeuralUCB (referred to as SupNeuralUCB). The Sup variant is an adoption of SupLinUCB, which was initially introduced in [69] for linear contextual bandits, to the neural setting. The authors in [162] proved an  $\tilde{O}(\sqrt{\Gamma_k(T)T})$  regret bound for SupNeuralUCB in the case of ReLU neural nets, which solved the issue of superlinear regret bound (non-diminishing instantaneous regret).

### 4.1.2 Contribution

All the existing works mentioned above, are limited to neural nets with ReLU activation functions. This limitation is rooted in the fact that the existing error bound between an overparameterized neural net and the corresponding NT kernel [13, Theorem 3.2] holds only for the ReLU neural nets. In Theorem 4.3.1, which may be of broader interest, we extend this result by providing error bounds for neural nets with arbitrarily smooth activation functions. Using Theorem 4.3.1, together with the recently established bounds on  $\Gamma_k(T)$  corresponding to arbitrarily smooth neural nets [300], we provide regret bounds

---

<sup>1</sup>The notations  $O$  and  $\tilde{O}$  are used for mathematical order and that up to polylogarithmic factors, respectively.

for neural contextual bandits under a more general setting than only ReLU activation function. In particular, in Theorem 4.4.1, we prove an  $\tilde{O}(T^{\frac{2d+2s-3}{2d+4s-4}})$  upper bound on regret, where  $s$  is the smoothness parameter of the activation functions ( $s = 1$  in the case of ReLU, and  $s$  grows larger as the activations become smoother). Our regret bounds recover that of [162] for ReLU activations, and can become as small as  $\tilde{O}(\sqrt{T})$  when the activations become infinitely smooth.

Broadening the scope of neural bandits to general smooth activations is of interest in two aspects. Firstly, as shown in [185, 218], deep neural nets constructed using smoother activation functions like Rectified Power Units or Re-PUs provide better approximation guarantees than those with ReLUs. This is particularly true when the function to be approximated is smooth, which is generally the case in practical applications. Additionally, smoother activation functions have also been shown to be more suitable for certain applications like implicit representations [271]. We explore whether such advantages of smoother activations reflect in the neural bandit settings. In particular, the regret bounds with ReLU neural nets, although sublinear, grow at a fast  $\tilde{O}(T^{\frac{2d-1}{2d}})$  rate, that quickly approaches the linear rate as  $d$  grows. We show that using neural nets with smoother activations can significantly reduce rate and thereby affirmatively establish the benefits of employing smoother activations in neural nets. Secondly, our results help establish connections between varying smoothness of activations in neural bandits and varying smoothness of kernels in kernel-based bandits. Kernel-based bandits [278] and Kernel-based contextual bandits [308] are well studied problems with regret bounds reported for popular kernels such as Matérn family, which is perhaps the most commonly used family of kernels in practice [273, 263]. Our regret bound for a neural contextual bandit problem with activations with smoothness parameter  $s$  is equivalent to the regret

bound in the case of kernel-based bandits with a Matérn kernel [308, 193] with smoothness parameter  $s - \frac{1}{2}$ . This connection may be of general interest in terms of contrasting neural nets and kernel-based models through NT kernel theory.

In Theorem 4.3.3, we prove confidence intervals for overparameterized neural nets with smooth activation functions, which are later used in the analysis of our algorithm. The confidence intervals are a consequence of our Theorem 4.3.1 and those for kernel regression. In contrast to prior work, our confidence intervals are tighter and hold for a general set of smooth activation functions (see Section 4.3).

In addition to the above analytical results for neural bandits with general smooth activation functions, we propose a practically efficient algorithm. The Sup variants of UCB algorithms are known to perform poorly in experiments [49, 193], including SupNeuralUCB presented in [162], due to over-exploration. We propose NeuralGCB, a neural contextual bandit algorithm guided by both upper and lower confidence bounds to provide a finer control of exploration-exploitation trade-off to avoid over-exploration. In the experiments, we show that NeuralGCB outperforms all other existing algorithms, namely NeuralUCB [346], NeuralTS [341] and SupNeuralUCB [162].

### 4.1.3 Other Related Work

Linear bandits have been considered also in a non-contextual setting, where the expected rewards are linear in the actions in a fixed domain, e.g., see the seminal work of Abbasi-Yadikori *et al.* [1]. The kernel-based bandits (also referred to as Gaussian process bandits) have been extensively studied under a

non-contextual setting [278, 68]. The GP-UCB and GP-TS algorithms proposed in this setting also show suboptimal regret bounds (see [305] for details). Recently there have been several works offering optimal order regret bounds for kernel-based (non-contextual) bandits [247, 52, 193]. These results however do not address the additional difficulties in the contextual setting.

## 4.2 Preliminaries and Problem Formulation

In the stochastic contextual bandit problem, at each discrete sequential step  $t = 1, 2, \dots, T$ , for each action  $a \in [K] := \{1, 2, \dots, K\}$ , a context vector  $\mathbf{x}_t \in \mathcal{X}$  is revealed, where  $\mathcal{X}$  is a compact subset of  $\mathbb{R}^d$ . Then, a bandit agent chooses an action  $a_t \in [K]$  and receives the corresponding reward  $r_t = h(\mathbf{x}_{t,a_t}) + \xi_t$ , where  $h : \mathbb{R}^d \rightarrow \mathbb{R}$  is the underlying reward function and  $\xi_t \in \mathbb{R}$  is a zero-mean martingale noise process. The goal is to choose actions which maximize the cumulative reward over  $T$  steps. This is equivalent to minimizing the cumulative *regret*, that is defined as the total difference between the maximum possible context-dependent reward and the actually received reward

$$R(T) = \sum_{t=1}^T (h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t})), \quad (4.1)$$

where  $a_t^* \in \arg \max_{a \in [K]} h(\mathbf{x}_{t,a})$  is the context-dependent maximizer of the reward function at step  $t$ . Following is a formal statement of the assumptions on  $\mathcal{X}$ ,  $f$  and  $\xi_t$ . These are mild assumptions which are shared with other works on neural bandits and NT kernel.

**Assumption 4.2.1.** (i) The input space is the  $d$  dimensional hypersphere:  $\mathcal{X} = \mathbb{S}^{d-1}$ . (ii) The reward function  $h$  belongs to the RKHS induced by the NT kernel,

and  $h(\mathbf{x}) \in [0, 1]$ ,  $\forall \mathbf{x} \in \mathcal{X}$ . (iii)  $\xi_t$  are assumed to be conditionally  $v$ -sub-Gaussian, i.e., for any  $\zeta \in \mathbb{R}$ ,  $\ln(\mathbb{E}[e^{\zeta \xi_t} | \xi_1, \dots, \xi_{t-1}]) \leq \zeta^2 v^2 / 2$ .

### 4.2.1 The Neural Net Model and Corresponding NT Kernel

In this section, we briefly outline the neural net model and the associated NT kernel. Let  $f(\mathbf{x}; \mathbf{W})$  be a fully connected feedforward neural net with  $L$  hidden layers of equal width  $m$ . We can express  $f$  using the following set of recursive equations:

$$\begin{aligned} f(\mathbf{x}; \mathbf{W}) &= \sqrt{\frac{c_s}{m}} \mathbf{W}^{(L+1)} \sigma_s(f^{(L)}(\mathbf{x})), \\ f^{(l)}(\mathbf{x}) &= \sqrt{\frac{c_s}{m}} \mathbf{W}^{(l)} \sigma_s(f^{(l-1)}(\mathbf{x})) \quad \text{for } 1 < l \leq L, \quad f^{(1)}(\mathbf{x}) = \mathbf{W}^{(1)} \mathbf{x} \end{aligned} \tag{4.2}$$

Let  $\mathbf{W} = (\mathbf{W}^{(l)})_{1 \leq l \leq L+1}$  denote the collection of all weights. The dimensions of weight matrices satisfy:  $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}$ ; for  $1 < l \leq L$ ,  $\mathbf{W}^{(l)} \in \mathbb{R}^{m \times m}$ ;  $\mathbf{W}^{(L+1)} \in \mathbb{R}^{1 \times m}$ . All the weights  $\mathbf{W}_{i,j}^{(l)}$   $\forall l, i, j$ , are initialized to  $\mathcal{N}(0, 1)$  and  $c_s := 2/(2s - 1)!!^2$ . With an abuse of notation,  $\sigma_s : \mathbb{R}^m \rightarrow \mathbb{R}^m$  is used for coordinate-wise application the activations of the form  $\sigma_s(u) = (\max(0, u))^s$  to the outputs of each layer. Note that  $\sigma_s$  is  $s - 1$  times differentiable everywhere which gives our results a general interpretability across smoothness of activations and the resulting function classes and allows us to draw parallels between our results and well established results for kernel-based bandits.

It has been shown that gradient based training of the neural net described above reaches a global minimum where the weights remain within a close vicinity of random initialization. As a result the model can be approximated with its

---

<sup>2</sup>For  $n \in \mathbb{N}$ , we define  $(2n - 1)!! = \prod_{i=1}^n (2i - 1)$ . For example  $3!! = 3$  and  $5!! = 15$ .

linear projection on the tangent space at random initialization  $\mathbf{W}_0$ :

$$f(\mathbf{x}; \mathbf{W}) \approx f(\mathbf{x}; \mathbf{W}_0) + (\mathbf{W} - \mathbf{W}_0)^\top \mathbf{g}(\mathbf{x}; \mathbf{W}_0),$$

where the notation  $\mathbf{g}(\mathbf{x}; \mathbf{W}) = \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W})$  is used for the gradient of  $f$  with respect to the parameters. The approximation error can be bounded by the second order term  $O(\|\mathbf{W} - \mathbf{W}_0\|^2)$ , and shown to be diminishing as the width  $m$  grows, where the implied constant depends on the spectral norm of the Hessian matrix [196]. The NT kernel corresponding to this neural net when  $m$  grows to infinity is defined as the kernel in the feature space determined by the gradient at initialization:

$$k(\mathbf{x}, \mathbf{x}') = \lim_{m \rightarrow \infty} \mathbf{g}^\top(\mathbf{x}; \mathbf{W}_0) \mathbf{g}(\mathbf{x}'; \mathbf{W}_0).$$

The particular form on the NT kernel depends on the activation function and the number of layers. For example, for a two layer neural net with ReLU activations, we have

$$k(\mathbf{x}, \mathbf{x}') = \frac{1}{\pi} \left( 2u(\pi - \arccos(u)) + \sqrt{1 - u^2} \right)$$

where  $u = \mathbf{x}^\top \mathbf{x}'$ . For the closed form derivation of NT kernel for other values of  $s$  and  $L$ , please refer to [300].

#### 4.2.2 Assumptions on Neural Net and NT Kernel

The following technical assumptions are mild assumptions which are used in the handling of the overparameterized neural nets using NT kernel. These assumptions are common in the literature and often are fulfilled without loss of generality [13, 346, 341, 120, 162].

**Assumption 4.2.2.** (i) Consider  $\mathbf{H} \in \mathbb{R}^{KT \times KT}$  such that  $[\mathbf{H}]_{i,j} = k(\mathbf{z}_i, \mathbf{z}_j)$  for all pairs in  $\mathcal{Z} \times \mathcal{Z}$ , where  $\mathcal{Z} = \{\{\mathbf{x}_{t,a}\}_{a=1}^K\}_{t=1}^T$ . We assume  $\mathbf{H} \succcurlyeq \lambda_0 \mathbf{I}$ , where  $\lambda_0 > 0$ . (ii)  $f(\mathbf{x}; \mathbf{W}_0)$  is assumed to be 0. This can always be ensured by an appropriate choice of initialization [346]. (iii) The number of epochs during training,  $J$ , and the learning rate,  $\eta$ , satisfy  $J = \frac{2}{c\lambda} T \log(T)$  and  $\eta = c'/T$  for some constants  $c, c' > 0$ .

### 4.2.3 Information Gain

The regret bounds for neural bandits are typically given in terms of maximal information gain (or the effective dimension) of the NT kernel. The maximal information gain is defined as the maximum log determinant of the kernel matrix [278]:

$$\Gamma_k(t) = \sup_{\mathbf{X}_t \subseteq \mathcal{X}} \frac{1}{2} \log \left( \det \left( \mathbf{I} + \frac{1}{\lambda} k_{\mathbf{X}_t, \mathbf{X}_t} \right) \right). \quad (4.3)$$

It is closely related to the effective dimension of the kernel, which denotes the number of features with a significant impact on the regression model and can be finite for a finite dataset even when the feature space of the kernel is infinite dimensional [340, 308]. It is defined as

$$\tilde{d}_k(t) = \text{trace} \left( k_{\mathbf{X}_t, \mathbf{X}_t} (k_{\mathbf{X}_t, \mathbf{X}_t} + \lambda \mathbf{I})^{-1} \right). \quad (4.4)$$

It is known that the information gain and the effective dimension are the same up to logarithmic factors [49]. We give our results in terms of information gain; nonetheless, that can be replaced by effective dimension.

### 4.3 Approximation Error and Confidence Bounds

As discussed earlier, the error between an overparameterized neural net and the associated NT kernel can be quantified and shown to be small for large  $m$ . To formalize this, we recall the technique of kernel ridge regression. Given a dataset  $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^t$ , let  $f_{\text{NTK}}$  denote the regressor obtained by kernel ridge regression using NT kernel. In particular, we have

$$f_{\text{NTK}}(\mathbf{x}) = k_{\mathbf{X}_t}^\top(\mathbf{x})(\lambda \mathbf{I}_t + k_{\mathbf{X}_t, \mathbf{X}_t})^{-1} \mathbf{Y}_t, \quad (4.5)$$

where  $k_{\mathbf{X}_t}(\mathbf{x}) = [k(\mathbf{x}_1, \mathbf{x}), \dots, k(\mathbf{x}_t, \mathbf{x})]^\top$  is the NT kernel evaluated between  $\mathbf{x}$  and  $t$  data points,  $k_{\mathbf{X}_t, \mathbf{X}_t} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^t$  is the NT kernel matrix evaluated on the data,  $\mathbf{Y}_t = [y_1, \dots, y_t]^\top$  is the vector of output values, and  $\lambda \geq 0$  is a free parameter. Note that  $f_{\text{NTK}}$  can be computed in closed form (without training a neural net). In addition, let  $f_{\text{NN}}$  be prediction of the neural net at the end of training using  $\mathcal{D}_t$ . Theorem 3.2 of Arora *et al.* [13] states that, when the activations are ReLU and  $m$  is sufficiently large, we have, with probability at least  $1 - \delta$ , that  $|f_{\text{NN}}(\mathbf{x}) - f_{\text{NTK}}(\mathbf{x})| \leq \epsilon$ . In this theorem, the value of  $m$  should be sufficiently large in terms of  $t, L, \delta, \epsilon$ , and  $\lambda_0$ . We now present a bound on the error between the neural net and the associated NT kernel for smooth activations.

**Theorem 4.3.1.** *Consider the neural net  $f(\mathbf{x}; \mathbf{W})$  defined in (4.2) consisting of  $L$  hidden layers each of width  $m \geq \text{poly}(1/\epsilon, s, 1/\lambda_0, |\mathcal{D}|, \log(1/\delta))$  with activations  $\sigma_s$ . Let  $f_{\text{NTK}}$  and  $f_{\text{NN}}$  be the kernel ridge regressor and trained neural net using a dataset  $\mathcal{D}$ . Then for any  $\mathbf{x} \in \mathcal{X}$ , with probability at least  $1 - \delta$  over the random initialization of the neural net, we have,*

$$|f_{\text{NTK}}(\mathbf{x}) - f_{\text{NN}}(\mathbf{x})| \leq \epsilon.$$

Theorem 4.3.1 is a generalization of Theorem 3.2 in Arora *et al.* [13] to the class of smooth activation functions  $\{\sigma_s, s \in \mathbb{N}\}$ . To prove Theorem 4.3.1, we show the following bound on the error between NT kernel and the kernel induced by the finite width neural net at initialization. This result is a generalization of Theorem 3.1 in Arora *et al.* [13] to smooth activation functions.

**Lemma 4.3.2.** *Consider the neural net  $f(\mathbf{x}; \mathbf{W})$  defined in Eqn. (4.2) consisting of  $L$  hidden layers each of width  $m \geq O(\frac{s^L L^2 c_s^2}{\varepsilon^2} \log^2(sL/\delta\varepsilon))$  with activations  $\sigma_s$ . Let  $\mathbf{g}(\mathbf{x}; \mathbf{W}) = \nabla_{\mathbf{W}} f(\mathbf{x}; \mathbf{W})$  and  $k$  be the associated NT kernel, as defined in Section 4.2.1. Fix  $\varepsilon > 0$  and  $\delta \in (0, 1)$ . Then for any  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ , with probability at least  $1 - \delta$  over the initialization of the network, we have,*

$$|\mathbf{g}^\top(\mathbf{x}; \mathbf{W}_0)\mathbf{g}(\mathbf{x}'; \mathbf{W}_0) - k(\mathbf{x}, \mathbf{x}')| \leq (L + 1)\varepsilon.$$

The proof entails a non-trivial generalization of various lemmas used in the proof of Theorem 3.2 in [13] to the class of smooth activation functions. We defer the detailed proof of the Lemma and the Theorem to Appendix C.1.

**Confidence Intervals:** The analysis of bandit problems classically builds on confidence intervals applicable to the values of the reward function. The NT kernel allows us to use the confidence intervals for kernel ridge regression, in building confidence intervals for overparameterized neural nets. In particular, given a dataset  $\mathcal{D}_t$ , let

$$\mathbf{V}_t = \lambda \mathbf{I} + \sum_{i=1}^t \mathbf{g}(\mathbf{x}_i; \mathbf{W}_0)\mathbf{g}^\top(\mathbf{x}_i; \mathbf{W}_0) \quad (4.6)$$

be the Gram matrix in the feature space determined by the gradient. Taking into account the linearity of the model in the feature space, we can define a

(surrogate) posterior variance<sup>3</sup> as follows,

$$\hat{\sigma}_t^2(\mathbf{x}) = \mathbf{g}^\top(\mathbf{x}) \mathbf{V}_t^{-1} \mathbf{g}(\mathbf{x}), \quad (4.7)$$

that can be used as a measure of uncertainty in the prediction provided by the trained neural net. Using Theorem 4.3.1 in conjunction with the confidence intervals for kernel ridge regression, we prove the following confidence intervals for a sufficiently wide neural net.

**Theorem 4.3.3.** *Let  $\mathcal{D}_t = \{(\mathbf{x}_i, r_i)\}_{i=1}^t$  denote a dataset obtained under the observation model described in Section 4.2 such that the points  $\{\mathbf{x}_i\}_{i=1}^t$  are independent of the noise sequence  $\{\xi_i\}_{i=1}^t$ . Suppose Assumptions 4.2.1 and 4.2.2 hold. Suppose the neural net defined in Eqn. (4.2) consisting of  $L$  hidden layers each of width  $m \geq \text{poly}(T, s, L, K, \lambda^{-1}, \lambda_0^{-1}, \log(1/\delta))$  is trained using this dataset. Then, with probability at least  $1 - \delta$ , the following relation holds for any  $\mathbf{x} \in \mathcal{X}$ :*

$$|h(\mathbf{x}) - f(\mathbf{x}; \mathbf{W}_t)| \leq \frac{C_{s,L} t}{\lambda m} + \beta_t \hat{\sigma}_t(\mathbf{x}), \quad (4.8)$$

where  $\mathbf{W}_t$  denotes the parameters of the trained model,  $\beta_t = S + 2\nu \sqrt{\log(1/\delta)} + C'_{s,L} t (1 - \eta \lambda)^{J/2} \sqrt{t/\lambda} + C''_{s,L} \sqrt{t^3/\lambda m}$ ,  $S$  is the RKHS (corresponding to the NT kernel) norm of  $h$  and  $C_{s,L}, C'_{s,L}, C''_{s,L}$  are constants depending only on  $s$  and  $L$ .

Both results presented in this section may be of broader interest in neural net literature. We would like to emphasize a couple of points here. Similar to [299, 162], the independence of query points from the noise sequence is fundamental to obtain these tighter bounds and hence cannot be directly used to improve the regret bounds of adaptive algorithms like NeuralUCB. Secondly, these bounds hold for all activation functions  $\{\sigma_s : s \in \mathbb{N}\}$  improving upon the existing results which hold only for ReLU activation function. Furthermore, this

---

<sup>3</sup>We use the notation  $\sigma(\cdot)$  to denote activation functions and  $\hat{\sigma}(\cdot)$  to denote posterior variance.

bound is tighter than the one derived in [162], even for the case of ReLU activation function. The confidence intervals are important building blocks in the analysis of NeuralGCB in Section 4.4. Please refer to Appendix C.3 for a detailed proof of the theorem.

## 4.4 Algorithm

The UCB family of algorithms achieve optimal regret in classic stochastic finite action bandits [15]. The optimal regret, however, hinges on the statistical independence of the actions. In more complex settings such as kernel-based and neural bandits, the inherent statistical dependence across adaptively chosen actions leads to skewed posterior distributions, resulting in sub-optimal and potentially trivial (i.e., superlinear) regret guarantees of UCB based algorithms [305]. To address this issue, one approach adopted in the literature is to use samples with limited adaptivity. These algorithms are typically referred to as Sup variant of UCB algorithms, and have been developed for linear [69], kernel-based [308] and neural [162] contextual bandits. These Sup variants, however, are known to perform poorly in practice due to their tendency to overly explore suboptimal arms.

We propose NeuralGCB, a neural bandit algorithm guided by both upper and lower confidence bounds to avoid over-exploring, leading to superior performance in practice while preserving the nearly optimal regret guarantee. The key idea of NeuralGCB is a finer control of the exploration-exploitation tradeoff based on the predictive standard deviation of past actions. This finer control encourages more exploitation and reduces unnecessary exploration. Specifically,

NeuralGCB partitions past action-reward pairs into  $R = \log T$  subsets (some subsets may be empty at the beginning of the learning horizon). Let  $\{\Psi^{(r)}\}_{r=1}^R$  denote these subsets of action-reward pairs where the index  $r$  represents the level of uncertainty about the data points in that set (with  $r = 1$  representing the highest uncertainty). Upon seeing a new context at time  $t$ , NeuralGCB traverses down the subsets starting from  $r = 1$ . At each level  $r$ , it evaluates the largest predictive standard deviation among current set of candidate actions,  $A_r$ , and compares it with  $2^{-r}$ .

If this value is smaller than  $2^{-r}$ , NeuralGCB checks if it can exploit based on predictions at the  $r^{\text{th}}$  level. Specifically, if the predictive standard deviation of the action that maximizes the UCB score is  $O(1/\sqrt{t})$  (indicating a high reward with reasonably high confidence), then NeuralGCB plays that action. Otherwise, it updates  $A_r$  by eliminating actions with small UCB score and moves to the next level. This encourages exploitation of less certain actions in a controlled manner as opposed to SupNeuralUCB which exploits only actions with much higher certainty.

On the other hand, if the value is greater than  $2^{-r}$ , NeuralGCB directly exploits the maximizer of the mean computed using data points in  $\Psi^{(r-1)}$  until the allocated budget for exploitation at level  $r$  is exhausted. It then resorts to more exploratory actions by choosing actions with large values of  $\hat{\sigma}_{t-1}^{(r)}$ . The exploitation budget is set to match the length of exploration sequence at the corresponding level to ensure an optimal balance of exploitation and exploration and preserve the optimal regret order while boosting empirical performance.

In addition to improved empirical performance, NeuralGCB takes into consideration the practical requirements for training the neural nets. Specifically,

---

**Algorithm 5** NeuralGCB

---

```

1: Require: Time horizon  $T$ , maximum initial variance  $\sigma_0$ , error probability  $\delta$ 
2: Initialize:  $R \leftarrow \lceil \log_2 T \rceil$ , Ensemble of  $R$  Neural Nets with  $\mathbf{W}_0^{(r)} = \mathbf{W}_0$ ,  $\Psi_0^{(r)} \leftarrow \emptyset$ ,  $\forall r \in [R]$ ,  $\mathcal{H} \leftarrow \emptyset$ , arrays  $\text{ctr}$ ,  $\text{max\_mu}$ ,  $\text{fb}$  of size  $R$  with all elements set to 0, batch sizes  $\{q_r\}_{r=1}^R$ 
3: for  $t = 1, 2, 3, \dots, T$  do
4:    $r \leftarrow 1, \hat{A}_r(t) = [K]$ 
5:   while True do
6:     Receive the context-action pairs  $\{\mathbf{x}_{t,a}\}_{a=1}^K$ 
7:      $\{f(\mathbf{x}_{t,a}; \mathbf{W}_t^{(r)}), \hat{\sigma}_{t-1}^{(r)}(\mathbf{x}_{t,a})\}_{a=1}^K, \mathbf{W}_t^{(r)}, \text{fb}[r] \leftarrow \text{GetPredictions}(\mathcal{H}, \Psi_{t-1}^{(r)}, \{\mathbf{x}_{t,a}\}_{a=1}^K, \text{fb}[r], \mathbf{W}_{t-1}^{(r)}, q_r)$ 
8:      $\tilde{\sigma}_{t-1}^{(r)} \leftarrow \max_{a \in \hat{A}_r(t)} \hat{\sigma}_{t-1}^{(r)}(x_{t,a})$ 
9:      $\text{max\_mu}[r] \leftarrow \arg \max_{a \in \hat{A}_r(t)} f(x_{t,a}; \mathbf{W}_{t-1}^{(r)})$ 
10:    if  $\tilde{\sigma}_{t-1}^{(r)} \leq \sigma_0 2^{-r}$  then
11:       $a_{\text{UCB}} \leftarrow \arg \max_{a \in \hat{A}_r(t)} \text{UCB}_{t-1}^{(r)}(\mathbf{x}_{t,a})$ 
12:      if  $\sigma_{t-1}^{(r)}(x_{t,a_{\text{UCB}}}) \leq \eta_0 / \sqrt{t}$  then
13:        Choose  $a_t \leftarrow a_{\text{UCB}}$  and set  $v_t \leftarrow 1$ 
14:        Receive  $y_t = h(\mathbf{x}_{t,a_t}) + \xi_t$  and update  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(x_{t,a_t}, y_t)\}$ 
15:        Set  $\Psi_t^{(r+1)} \leftarrow \Psi_{t-1}^{(r+1)} \cup \{(t, v_t)\}$  and  $\Psi_t^{(r')} \leftarrow \Psi_{t-1}^{(r')}$  for all  $r' \in [R] \setminus \{r+1\}$ 
16:        break
17:      else
18:         $\hat{A}_{r+1}(t) \leftarrow \{a \in \hat{A}_r(t) : \text{UCB}_{t-1}^{(r)}(\mathbf{x}_{t,a}) \geq \max_{a' \in \hat{A}_r(t)} \text{LCB}_{t-1}^{(r)}(\mathbf{x}_{t,a'})\}, r \leftarrow r + 1$ 
19:      end if
20:    else
21:      if  $r = 1$  or  $\text{ctr}[r] > \alpha_0 4^r$  then
22:        Choose any  $a_t \in \hat{A}_r(t)$  such that  $\hat{\sigma}_{t-1}^{(r)}(x_{t,a}) > \sigma_0 2^{-r}$  and set  $v_t \leftarrow 2$ 
23:      else
24:        Choose  $a_t \leftarrow \text{max\_mu}[r-1]$  and set  $v_t \leftarrow 3$ 
25:      end if
26:      Receive  $y_t = h(\mathbf{x}_{t,a_t}) + \xi_t$  and update  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(x_{t,a_t}, y_t)\}, \text{ctr}[r] \leftarrow \text{ctr}[r] + 1$ 
27:      Set  $\Psi_t^{(r)} \leftarrow \Psi_{t-1}^{(r)} \cup \{(t, v_t)\}$  and  $\Psi_t^{(r')} \leftarrow \Psi_{t-1}^{(r')}$  for all  $r' \in [R] \setminus \{r\}$ 
28:      break
29:    end if
30:  end while
31: end for

```

---

as pointed out in [120], it is practically more efficient to train the neural net over batches of observations, rather than sequentially at each step. Consequently, before evaluating the mean and variance at any level  $r$ , NeuralGCB retrains the neural net corresponding to that level only if  $q_r$  samples have been added to

that level since it was last trained. This index-dependent choice of batch size,  $q_r$  lends a natural adaptivity to the retraining frequency by reducing it as time progresses.

A pseudo code of the algorithm is outlined in Algorithm 5. In the pseudo code,  $\text{UCB}_t^{(r)}$  and  $\text{LCB}_t^{(r)}$  refer to the upper and lower confidence scores respectively at time  $t$  corresponding to index  $r$  and are defined as  $\text{UCB}_t^{(r)}(\cdot) = f(\cdot; \mathbf{W}_t^{(r)}) + \beta\hat{\sigma}_t^{(r)}(\cdot)$  and  $\text{LCB}_t^{(r)}(\cdot) = f(\cdot; \mathbf{W}_t^{(r)}) - \beta\hat{\sigma}_t^{(r)}(\cdot)$ . `GetPredictions` is a local routine that calculates the predictive mean and variance after appropriately retraining the neural net (see Appendix C.4 for a pseudo code). Lastly, the arrays `ctr`, `max_mu` and `fb` are used to store the exploitation count, maximizer of the neural net output and feedback time instants. We now formally state the regret guarantees for NeuralGCB in the following theorem.

**Theorem 4.4.1.** *Suppose Assumptions 4.2.1 and 4.2.2 hold. Consider NeuralGCB given in Algorithm 5, with  $R$  neural nets, as defined in Eqn. (4.2) with  $L$  hidden layers each of width  $m \geq \text{poly}(T, L, K, \lambda^{-1}, \lambda_0^{-1}, S^{-1}, \log(1/\delta))$ . Suppose NeuralGCB is run with a fixed batch size for each group, then the regret defined in Eqn. (4.1) satisfies*

$$R(T) = \tilde{O}\left(\sqrt{T\Gamma_k(T)} + \sqrt{T\Gamma_k(T)\log(1/\delta)} + \max_r q_r\Gamma_k(T)\right)$$

As suggested by the above theorem, NeuralGCB preserves the regret guarantees of SupNeuralUCB which are much tighter than those of NeuralUCB. Moreover, these guarantees hold for even for smooth activation functions as opposed to just for ReLU activation. Furthermore, the above regret guarantees show that the retraining neural nets only at the end of batches of observations increases the regret at most with an additive term in the batch size. Our results also hold with an adaptive batch size. Please refer to Appendix C.4 for a detailed proof of the Theorem and additional details about NeuralGCB.

## 4.5 Empirical Studies

In this section, we provide numerical experiments on comparing NeuralGCB with several representative baselines, namely, LinUCB [69], NeuralUCB [346], NeuralTS [341], SupNeuralUCB [162] and Batched NeuralUCB [120].

We perform the empirical studies on three synthetic and two real-world datasets. We first compare NeuralGCB with the fully sequential algorithms (LinUCB, NeuralUCB, NeuralTS and SupNeuralUCB). We then compare the regret incurred and the time taken by NeuralGCB, NeuralUCB and Batch NeuralUCB. We perform both set of experiments with two different activation functions. The construction of the synthetic datasets and the experimental settings are described below.

### 4.5.1 Datasets

For each of the synthetic datasets, we construct a contextual bandit problem with a feature dimension of  $d = 10$  and  $K = 4$  actions per context running over a time horizon of  $T = 2000$  rounds. The set of context vectors  $\{\{\mathbf{x}_{t,a}\}_{a=1}^K\}_{t=1}^T$  are drawn uniformly from the unit sphere. Similar to Zhou *et al.* [346], we consider the following three reward functions:

$$h_1(\mathbf{x}) = 4|\mathbf{a}^\top \mathbf{x}|^2; \quad h_2(\mathbf{x}) = 4 \sin^2(\mathbf{a}^\top \mathbf{x}); \quad h_3(\mathbf{x}) = \|\mathbf{A}\mathbf{x}\|_2. \quad (4.9)$$

For the above functions, the vector  $\mathbf{a}$  is drawn uniformly from the unit sphere and each entry of matrix  $\mathbf{A}$  is randomly generated from  $\mathcal{N}(0, 0.25)$ .

We also consider two real datasets for classification namely Mushroom and

Statlog (Shuttle), both of which are available on the UCI repository [100].

**Mushroom:** The Mushroom dataset consists of 8124 samples with 22 features where each data point is labelled as an edible or poisonous mushroom, resulting in a binary classification problem. For the purpose of the experiments, we carry out some basic preprocessing on this dataset. We drop the attributed labeled “veil-type” as it is the same for all the datapoints resulting in a  $d = 21$  dimensional feature vector. Furthermore, we randomly select 1000 points from each class to create a smaller dataset of size 2000 to run the experiments.

**Statlog:** The Statlog (Shuttle) dataset consists of 58000 entries with  $d = 7$  attributes (we do not consider time as an attribute) each. The original dataset is divided into 7 classes. Since the data is skewed, we convert this into a binary classification problem by combining classes. In particular, we combine the five smallest classes, namely, 2, 3, 5, 6, 7 and combine them to form a single class and drop class 4 resulting in a binary classification problem, with points labelled as 1 forming the first class and ones with labels in {2, 3, 5, 6, 7} forming the second. Once again, we select a subset of 2000 points by randomly choosing 1000 points from each class to use for the experiments. Lastly, we normalize the features of this dataset.

The classification problem is then converted into a contextual bandit problem using techniques outlined in Li *et al.* [188], which is also adopted in [346] and [120]. Specifically, each datapoint in the dataset  $(\mathbf{x}, y) \in \mathbb{R}^d \times \mathbb{R}$  is transformed into  $K$  vectors of the form  $\mathbf{x}^{(1)} = (\mathbf{x}, \mathbf{0}, \dots, \mathbf{0}) \dots, \mathbf{x}^{(K)} = (\mathbf{0}, \dots, \mathbf{0}, \mathbf{x}) \in \mathbb{R}^{Kd}$  corresponding to the  $K$  actions associated with context  $\mathbf{x}$ . Here  $K$  denotes the

number of classes in the original classification problem. The reward function is set to  $h(\mathbf{x}^{(k)}) = \mathbb{1}\{y = k\}$ , that is, the agent receives a reward of 1 if they classify the context correctly and 0 otherwise. As mentioned above, we have  $d = 21$  for Mushroom dataset and  $d = 7$  for the Shuttle dataset and  $K = 2$  for both of them.

### 4.5.2 Experimental Setting

For all the experiments, the rewards are generated by adding zero mean Gaussian noise with a standard deviation of 0.1 to the reward function. All the experiments are run for a time horizon of  $T = 2000$ . We report the regret averaged over 10 Monte Carlo runs with different random seeds along with an error bar of one standard deviation on either side. For all the datasets, we generate new rewards for each Monte Carlo run by changing the noise. Furthermore, for the real datasets, we also shuffle the context vectors for each Monte Carlo run.

**Setting the hyperparameters:** For all the algorithms, the parameter  $\nu$ , the standard deviation proxy for the sub-Gaussian noise, to 0.1, the standard deviation of the Gaussian noise added. Also, the RKHS norm of the reward,  $S$  to 4 for synthetic functions, and 1 for real datasets in all the algorithms. For all the experiments, we perform a grid search for  $\lambda$  and  $\eta$  over  $\{0.05, 0.1, 0.5\}$  and  $\{0.001, 0.01, 0.1\}$ , respectively, for each algorithm and choose the best ones for the corresponding algorithm and reward function. The exploration parameter  $\beta_t$  is set to the value prescribed by each algorithm. For NeuralUCB and NeuralTS, the prescribed value for the exploration parameter in the original work

was given as:

$$\begin{aligned}\beta_t = \beta_0 & \left( \nu \sqrt{\log \left( \frac{\det(\mathbf{V}_t)}{\det(\lambda \mathbf{I})} \right) + C_2 m^{-1/6} \sqrt{\log m} L^4 t^{5/3} \lambda^{-1/6} + 2 \log(1/\delta) + S} \right) \\ & + (\lambda + C_3 t L) \left[ (1 - \eta m \lambda)^{J/2} \sqrt{t/\lambda} + m^{-1/6} \sqrt{\log m} L^{7/2} t^{5/3} \lambda^{-5/3} (1 + \sqrt{t/\lambda}) \right],\end{aligned}$$

where  $\beta_0 = \sqrt{1 + C_1 m^{-1/6} \sqrt{\log m} L^4 t^{7/6} \lambda^{-7/6}}$ . We ignore the smaller order terms and set  $\beta_t = 2S + \nu \sqrt{\log \left( \frac{\det(\mathbf{V}_t)}{\det(\lambda \mathbf{I})} \right) + 2 \log(1/\delta)}$  as  $(\lambda + C_3 t L)(1 - \eta m \lambda)^{J/2} \sqrt{t/\lambda} \leq S$  for given choice of parameters as shown in Zhou *et al.* [346]. For NeuralGCB and SupNNUCB, we apply the same process of ignoring the smaller order terms and bounding the additional constant term by  $S$  to obtain  $\beta_t = 2S + \nu \sqrt{\frac{2}{\lambda} \log(1/\delta)}$ , which is used in the algorithms. Also for NeuralGCB,  $\eta_0$  was set to 0.2 and  $\alpha_0$  to  $20 \log T$ , as suggested in Appendix C.4, for all experiments. The number of epochs is set to 200 for synthetic datasets and Mushroom, and to 400 for Statlog.

**Neural Net architecture:** We consider a 2 layered neural net as described in Eqn. (4.2) for all the experiments. We perform two different sets of experiments with two different activation functions, namely  $\sigma_1$  (or equivalently, ReLU) and  $\sigma_2$ . For the experiments with  $\sigma_1$  as the activation, we set the number of hidden neurons to  $m = 20$  and  $m = 50$  for synthetic and real datasets respectively. For those with  $\sigma_2$ ,  $m$  is set to 30 and 80 for synthetic and real datasets, respectively.

**Training Schedule:** For the experiments with sequential algorithms, we retrain the neural nets at every step, including NeuralGCB. For the batched algorithms, we consider a variety of training schedules. Particularly, for each setting (i.e., dataset and activation function) we consider two fixed and two adaptive batch size training schedules, which are denoted by  $F1, F2$  and  $A1, A2$  respectively. We specify the training schedules below for different dataset and activa-

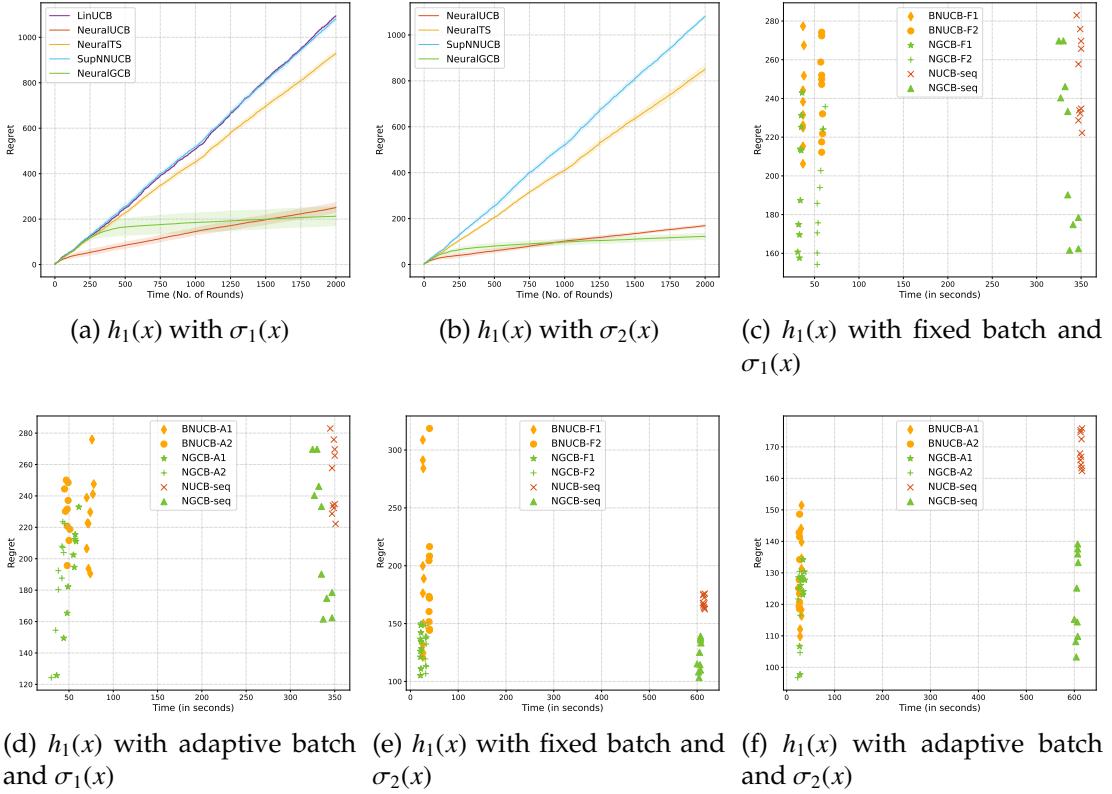


Figure 4.1: Plots with reward function  $h_1(x)$

tion functions beginning with those for  $\sigma_1$  (ReLU). These were chosen to ensure that both algorithms have roughly the same running time to allow for a fair comparison.

1. For Batched NeuralUCB run on a synthetic function:
  - $F1$ : 200 batches, or equivalently retrain after every 10 steps.
  - $F2$ : 300 batches, or equivalently retrain after every 6 – 7 steps.
  - $A1$ :  $q = 2$
  - $A2$ :  $q = 3$
2. For Batched NeuralUCB run on Mushroom:
  - $F1$ : 100 batches, or equivalently retrain after every 20 steps.

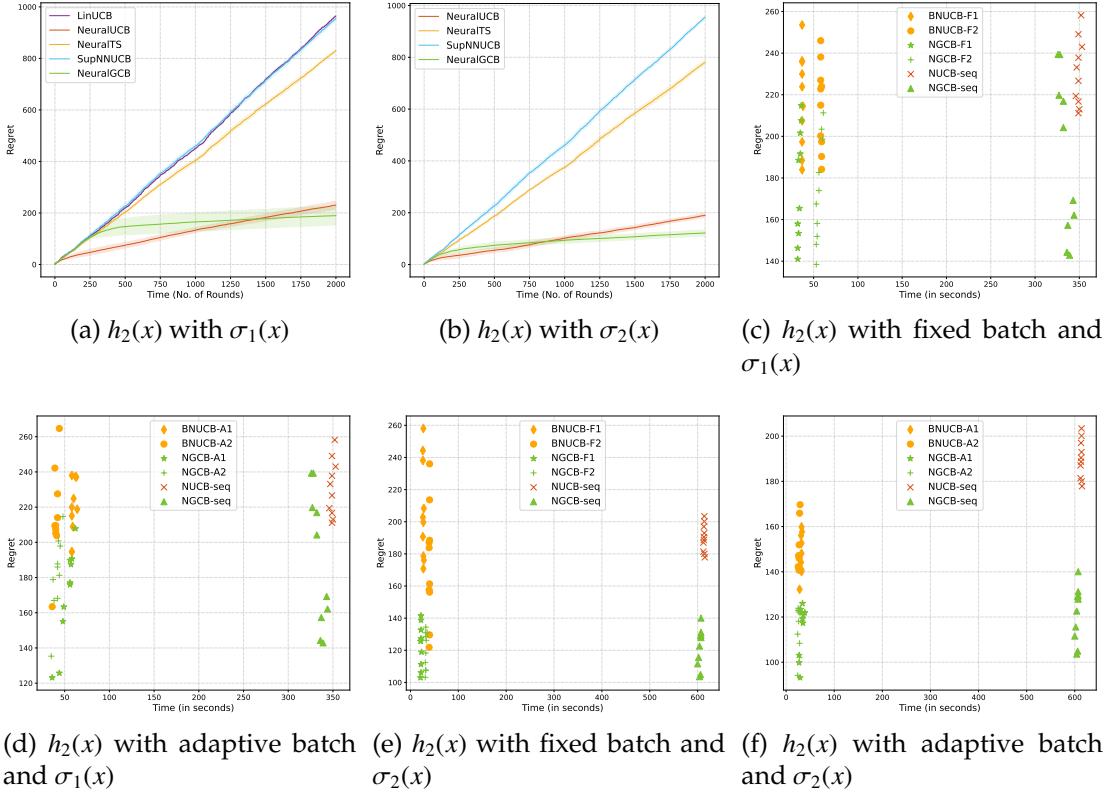


Figure 4.2: Plots with reward function  $h_2(x)$

- $F2$ : 200 batches, or equivalently retrain after every 10 steps.
- $A1$ :  $q = 500$
- $A2$ :  $q = 700$

### 3. For Batched NeuralUCB run on Shuttle:

- $F1$ : 100 batches, or equivalently retrain after every 20 steps.
- $F2$ : 200 batches, or equivalently retrain after every 10 steps.
- $A1$ :  $q = 5$
- $A2$ :  $q = 10$

### 4. For NeuralGCB run on a synthetic function:

- $F1$ :  $q_1 = 4, q_2 = 20, q_r = 40$  for  $r \geq 3$  (i.e., retrain after  $q_r$  steps)

- $F2$ :  $q_1 = 2, q_2 = 10, q_r = 20$  for  $r \geq 3$ .
- $A1$ :  $q_1 = 1.2, q_2 = 1.8, q_r = 2.7$  for  $r \geq 3$ .
- $A2$ :  $q_1 = 1.5, q_2 = 2.25, q_r = 3.375$  for  $r \geq 3$ .

5. For NeuralGCB run on Mushroom:

- $F1$ :  $q_1 = 20, q_2 = 40, q_r = 80$  for  $r \geq 3$  (i.e., retrain after  $q_r$  steps)
- $F2$ :  $q_1 = 10, q_2 = 20, q_r = 40$  for  $r \geq 3$ .
- $A1$ :  $q_1 = 300, q_2 = 1200, q_r = 4800$  for  $r \geq 3$ .
- $A2$ :  $q_1 = 500, q_2 = 2000, q_r = 8000$  for  $r \geq 3$ .

6. For NeuralGCB run on Shuttle:

- $F1$ :  $q_1 = 20, q_2 = 40, q_r = 80$  for  $r \geq 3$  (i.e., retrain after  $q_r$  steps)
- $F2$ :  $q_1 = 10, q_2 = 20, q_r = 40$  for  $r \geq 3$ .
- $A1$ :  $q_1 = 5, q_2 = 12.5, q_r = 31.25$  for  $r \geq 3$ .
- $A2$ :  $q_1 = 10, q_2 = 25, q_r = 62.5$  for  $r \geq 3$ .

Similarly, for  $\sigma_2$ , the training schedules are given as follows:

1. For Batched NeuralUCB run on a synthetic function:

- $F1$ : 200 batches, or equivalently retrain after every 10 steps.
- $F2$ : 300 batches, or equivalently retrain after every 6 – 7 steps.
- $A1$ :  $q = 3$
- $A2$ :  $q = 4$

2. For Batched NeuralUCB run on Mushroom:

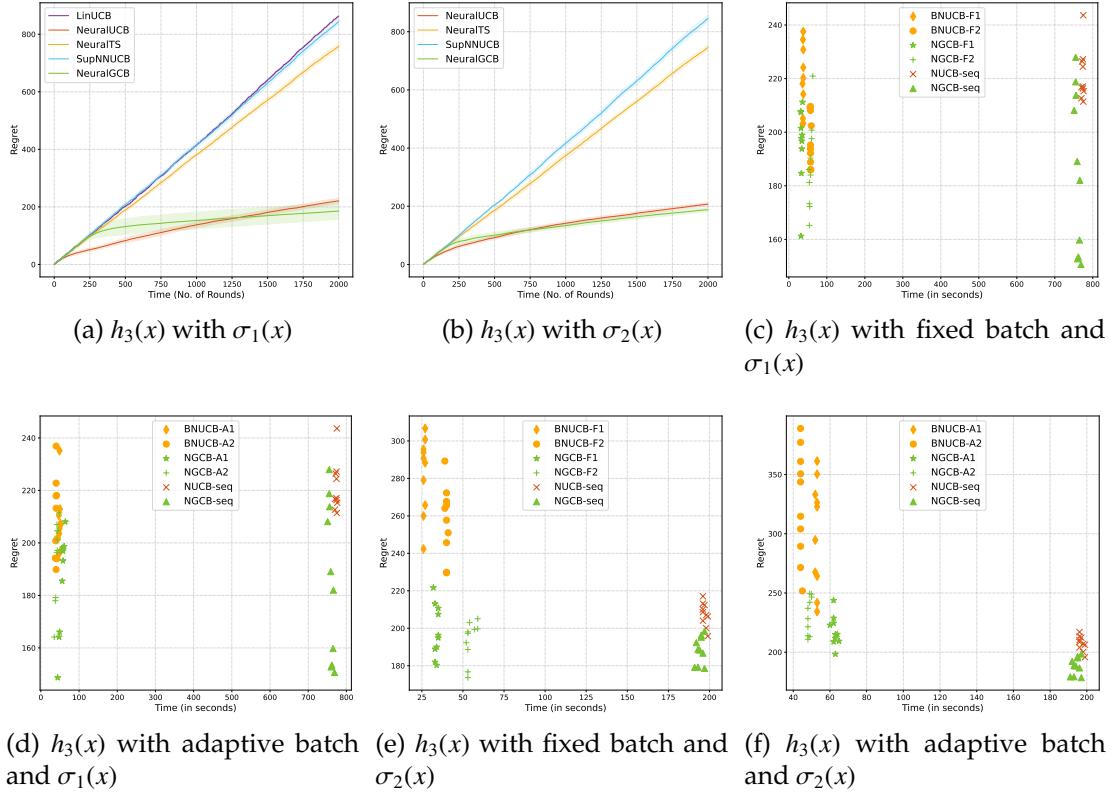


Figure 4.3: Plots with reward function  $h_3(x)$

- $F1$ : 100 batches, or equivalently retrain after every 20 steps.
- $F2$ : 200 batches, or equivalently retrain after every 10 steps.
- $A1$ :  $q = 500$
- $A2$ :  $q = 700$

### 3. For Batched NeuralUCB run on Shuttle:

- $F1$ : 50 batches, or equivalently retrain after every 40 steps.
- $F2$ : 100 batches, or equivalently retrain after every 20 steps.
- $A1$ :  $q = 5$
- $A2$ :  $q = 10$

### 4. For NeuralGCB run on a synthetic function:

- $F1: q_1 = 4, q_2 = 20, q_r = 40$  for  $r \geq 3$  (i.e., retrain after  $q_r$  steps)
- $F2: q_1 = 2, q_2 = 10, q_r = 20$  for  $r \geq 3$ .
- $A1: q_1 = 1.2, q_2 = 1.8, q_r = 2.7$  for  $r \geq 3$ .
- $A2: q_1 = 1.5, q_2 = 2.25, q_r = 3.375$  for  $r \geq 3$ .

5. For NeuralGCB run on Mushroom:

- $F1: q_1 = 10, q_2 = 30, q_r = 60$  for  $r \geq 3$  (i.e., retrain after  $q_r$  steps)
- $F2: q_1 = 10, q_2 = 20, q_r = 40$  for  $r \geq 3$ .
- $A1: q_1 = 300, q_2 = 1200, q_r = 4800$  for  $r \geq 3$ .
- $A2: q_1 = 500, q_2 = 2000, q_r = 8000$  for  $r \geq 3$ .

6. For NeuralGCB run on Shuttle:

- $F1: q_1 = 5, q_2 = 12, q_r = 31$  for  $r \geq 3$  (i.e., retrain after  $q_r$  steps)
- $F2: q_1 = 5, q_2 = 10, q_r = 20$  for  $r \geq 3$  (i.e., retrain after  $q_r$  steps)
- $A1: q_r = 3$  for all  $r$
- $A2: q_r = 4$  for all  $r$

We use this notation of  $F1, F2, A1$  and  $A2$  to refer to the training schedules in the plots shown in the next section.

### 4.5.3 Results

In this section, we plot the performance of NeuralGCB against several baselines for different experimental setups and reward functions. For each reward function, we plot the following 6 figures:

- Fig. (a): We plot the cumulative regret against time (number of rounds) for various baselines for the case where the activation function of the underlying neural net is  $\sigma_1$ , or equivalently, ReLU.
- Fig (b): We plot the cumulative regret against time (number of rounds) for various baselines for the case where the activation function of the underlying neural net is  $\sigma_2$ .
- Fig (c)-(f): We compare the regret and running time (in seconds) between the batched and the sequential versions of NeuralUCB and NeuralGCB. We plot the regret incurred against time taken (running time, in seconds) for Batched-NeuralUCB (BNUCB), (Batched)-NeuralGCB (BNGCB), NeuralUCB (NUCB-seq), (Sequential) NeuralGCB (NGCB-seq) under different training schedules for 10 different runs. In particular, in (c) and (e), we consider fixed batch sizes with  $\sigma_1$  and  $\sigma_2$  as activation functions respectively and in (d) and (f) with consider adaptive batch sizes with  $\sigma_1$  and  $\sigma_2$  as activation functions respectively. The batch sizes used in the plots are described in the previous section.

From the figures, it is evident that NeuralGCB outperforms all other existing algorithms on variety of datasets, including both synthetic and real world datasets. Moreover, the conclusion holds equally well for different activation functions, further bolstering the practical efficiency of the proposed algorithm. Additionally, it can be noted that the regret incurred by the algorithms in experiments with  $\sigma_2$  as the activation is less than that for the experiments with  $\sigma_1$  as the activation, thereby demonstrating the effect of smooth kernels on the performance of the algorithms in practice. We would like to point out that we only focus on algorithms with theoretical guarantees in this study. There are various approximate TS algorithms that also have good empirical performance [238]

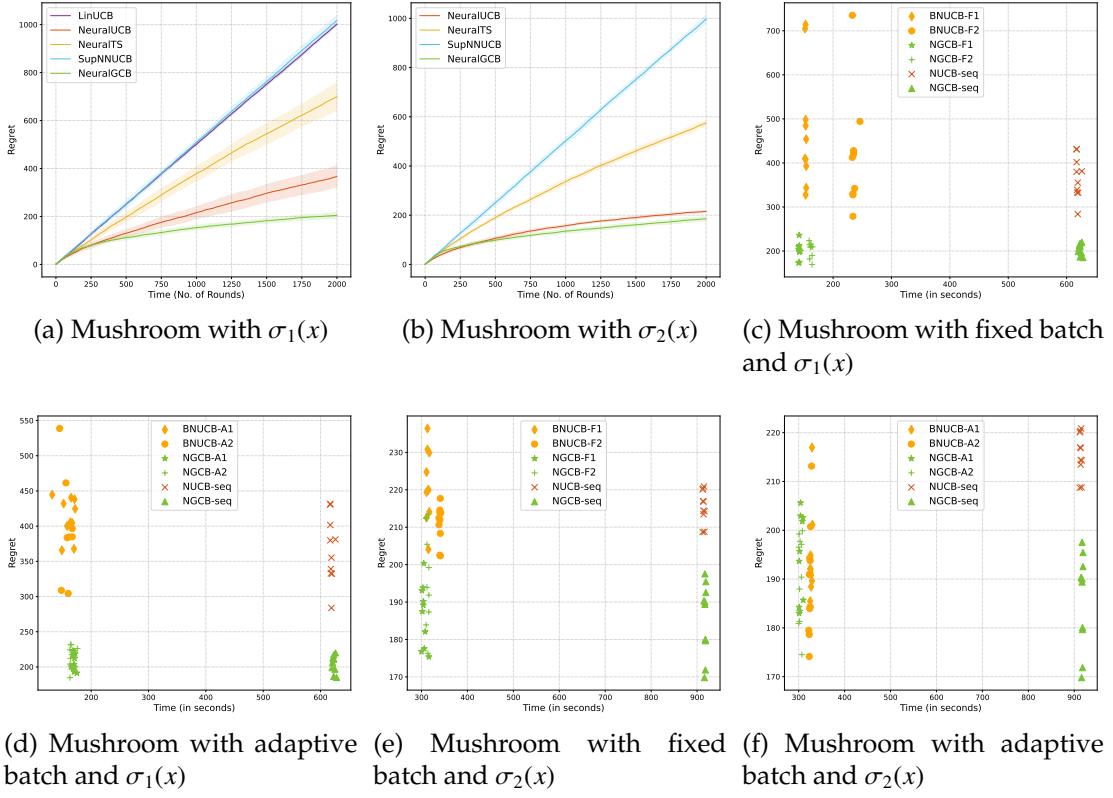


Figure 4.4: Plots for the dataset Mushroom

but lack such provable efficiency, which is central to the motivation of our paper. In addition, the formulation of contextual bandits in [238] is different from the usual one in the literature [308, 346, 341, 120, 162], considered in our work, which also makes a direct empirical comparison infeasible.

Lastly, the extensive study with different training schedules shows that batched version performs almost as well as the fully sequential version in terms of regret on all the datasets while having a considerably smaller running time. It can be noted from the plots that NeuralGCB has a superior performance compared to Batched-NeuralUCB in terms of regret for comparable running times on different datasets and with different activation functions which further strengthens the case of NeuralGCB as a practically efficient algorithm.

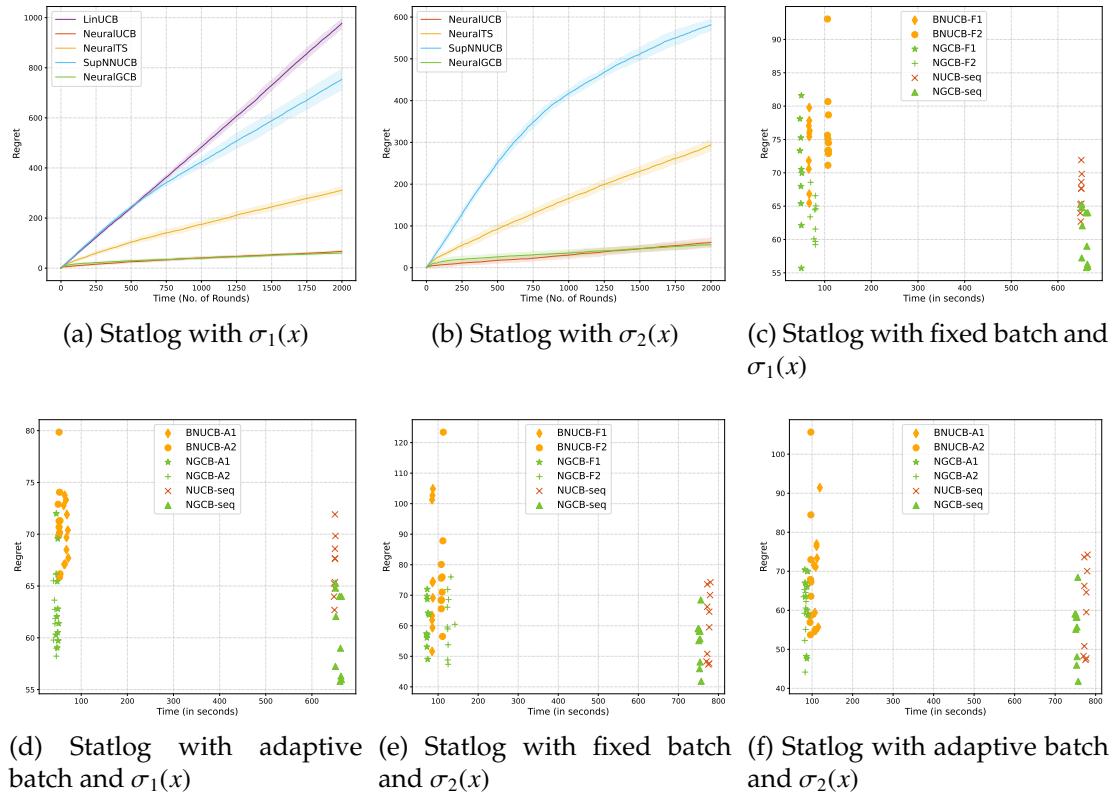


Figure 4.5: Plots for the dataset Statlog (Shuttle)

## **Part II**

# **Distributed Stochastic Optimization and Learning**

CHAPTER 5

**COMMUNICATION-EFFICIENT LEARNING FOR DISTRIBUTED  
STOCHASTIC OPTIMIZATION**

## 5.1 Introduction

The tension between learning efficiency and communication cost is evident in distributed optimization, as pointed out earlier in Chapter 1. If distributed agents (or clients) can share all their locally obtained information and fully coordinate their actions, the problem effectively reduces to a centralized problem, and the greatest learning efficiency defined by the centralized counterpart is trivially achieved at the price of high communication cost.

What is the minimum amount (in terms of bits) of communications needed to achieve the learning efficiency offered by centralized learning? How one might design a distributed learning algorithm that operates at such an optimal point in the communication-learning efficiency trade-off curve? These fundamental questions have not been adequately addressed in the literature.

### 5.1.1 Main Results

In this chapter, we address the above questions within the scope of distributed linear bandits, or equivalently, distributed zeroth-order optimization of linear functions. We consider a system of  $M$  distributed agents whose actions generate random rewards governed by a common unknown mean  $\theta^* \in \mathbb{R}^d$ . The agents aim to optimize their actions over time to minimize the overall cumulative re-

gret incurred by all agents over a horizon of length  $T$ . Communications across agents are facilitated by a central server. To quantify the communication cost to the bit level, we assume that both the uplink and downlink channels have a capacity of  $R$  bits per channel use.

Our main results are twofold. First, we establish an information-theoretic lower bound on the required communications for achieving a sublinear regret order. Second, we develop an efficient algorithm that achieves the optimal regret order offered by centralized learning using the minimum order of communications dictated by the information-theoretic lower bound. For sparse linear bandits, we show a variant of the proposed algorithm offers better regret-communication trade-off by leveraging the sparsity of the problem.

For the distributed linear bandit problem, to achieve the optimal regret order of  $\Omega(d \sqrt{MT})$  in both  $M$  and  $T$  as offered by centralized learning, the agents need to cooperate in learning the underlying reward vector  $\theta^*$ . In addition to a policy for choosing reward-generating actions at each time, a distributed learning algorithm also includes a communication strategy that governs *when* to communicate and *what* and *how* to send it (i.e., quantization and encoding) over the finite-capacity channels. To minimize the total regret that is accumulating over time and aggregating over the agents while using a minimum amount of communications, the communication strategy needs to work in tandem with action selection to ensure a continual flow of information available at all agent for decision-making.

The key idea of the proposed algorithm is an *progressive learning and sharing* (PLS) structure that systematically coordinates the collective exploration of  $\theta^*$  and the information sharing of the estimates of  $\theta^*$  over finite-capacity chan-

nels. Specifically, the PLS algorithm progresses as each agent learns and shares one-bit information about  $\theta^*$  (per dimension) at a time, starting from the most significant bit in the binary expansion of  $\theta^*$  in each dimension. This bit-by-bit learning and sharing is structured in interleaving exploration and exploitation epochs with carefully controlled epoch lengths, to achieve both the minimum order of channel usage and the minimum order of cumulative regret.

### 5.1.2 Related Work

#### Communication Efficiency in Distributed Bandits

Communication cost is commonly partitioned into two parts: the size of the message at each information exchange and the frequency of information exchange. As detailed below, these two sides of the same coin have largely been dealt with separately in the literature when developing communication-efficient algorithms for distributed bandit problems. Our work departs from existing studies by taking a holistic view on communication cost and making an initial attempt at characterizing the information-theoretic limit on the communication-learning trade-off in distributed linear bandits.

In the group of work focusing on reducing communication frequency through intermittent information exchange, it is often assumed that the information being transmitted, typically a vector in  $\mathbb{R}^d$ , can be communicated with infinite precision, which requires a channel with infinite capacity. There have been numerous studies in both distributed multi-armed bandits and distributed linear bandits that adopt this approach towards achieving communication efficiency.

The multi-armed bandit (MAB) problem with a finite number of arms has been extensively studied in various distributed learning setups. A line of work [64, 176, 262, 171, 254, 211, 269, 347] focuses on developing algorithms for cooperative learning in MAB under different structures of the underlying network. Another line of work [198, 153, 240, 36] explores a collision based approach with no explicit communication inspired by cognitive radio networks. There are several works that also consider the impact of communication and develop communication efficient algorithms by either reducing the frequency of communication [8, 136, 285] or proposing quantization approaches [123]. The setup of linear bandits considered in this work is more challenging than the MAB setup considered in the above works, especially for the communication constrained setting.

The problem of distributed linear bandits has also received considerable attention in the literature. Ghosh *et al.* [115] study the problem under heterogeneity assumptions on the agents and propose a new algorithms under personalization and clustering frameworks, the two different ways adopted to tackle heterogeneity. Chawla *et al.* [65] consider a high dimensional linear bandit setting where the underlying mean reward lies in a low-dimensional space chosen from a known, finite collection. They propose a decentralized algorithm based on communication over a network to quickly identify this subspace to ensure sub-linear regret. While there is some focus on communication in this study, the proposed algorithm does not offer a linear speed up with respect to the number of agents, although per agent regret improves under collaboration. Korda *et al.* [171] consider the problem in a peer-to-peer communication model instead of a star topology. They propose an algorithm that achieves order-optimal regret guarantees with limited communication. However, their proposed policy

requires communication of  $O(d^2)$  bits per message over the network which is worse than the  $O(d)$  required by our proposed algorithm. Under the same setting as considered in our work, Wang *et al.* [320] proposed an algorithm based on batched elimination of arms where poorly performing arms are eliminated at the end of each batch. The batched structure offers a natural way to limit communication to one message exchange per batch. Recently, Huang *et al.* [143] and Amani *et al.* [11] have extended the batch elimination idea to the settings of heterogenous agents and stochastic contextual linear bandits. However, there is no constraint on the amount of information that can be transmitted in each batch, allowing numbers to be sent with infinite precision.

The other group of work focuses on reducing the size of the message at each information exchange. The frequency of communication is not a concern. The objective is to best *approximate* the information being exchanged via techniques such as quantization and sparsification (see, for example, [169, 123, 212, 282]). In particular, Hanna *et al.* [123] proposed a quantization scheme to reduce the communication overhead in discrete multi-armed bandit problems at the cost of a small multiplicative constant in the regret. Recently, Mitra *et al.* [212] proposed an algorithm for decentralized linear bandits with a finite-capacity uplink channel and an infinite-capacity downlink channel. They developed an adaptive encoding scheme for communication that ensured order-optimal regret for their proposed algorithm. However, with a linear order of message exchanges in  $T$ , the total uplink communication cost is  $O(dT)$  bits as opposed to the  $O(d \log T)$  bits of communication cost in our proposed algorithm. Moreover, the results in [210] hold only for single agent while ours hold for a distributed setup with multiple agents.

## Sparse Linear Bandits

The problem of sparse linear bandits has been studied mainly in the centralized setting. Abbasi-Yadikori *et al.* [2] proposed an algorithm for sparse linear bandits using a online to batch conversion of the confidence intervals. Borrowing techniques from compressed sensing, Carpentier and Munos [57] proposed an algorithm that incurs an overall regret of  $O(s\sqrt{T})$ . Chen *et al.* [67] proposed the sparse variants of the famous Lin-UCB [1] and Sup-Lin-UCB [69] algorithms for the contextual bandit problem. Hao *et al.* [132] proposed an algorithm with dimension-independent regret bounds for very high dimensional problems where the dimension is much larger than the time horizon. As mentioned previously, all these studies consider the centralized setting which is different from the distributed setting considered in this work. To the best of our knowledge, this is the first work considering sparse linear bandits under a distributed setting with communication constraints.

## Lower Bounds for Distributed Bandits

Several studies have attempted to characterize information theoretic lower bounds on communication under for various statistical estimation tasks. The classical paper Duchi *et al.* [103] derived guarantees on communication requirements for distributed mean estimation for both independent and interactive protocols. Similarly, Tsitsiklis and Luo [297] have studied the communication complexity in convex optimization while Diakonikolas *et al.* [94] study the effect of communication constraints for the task of distribution estimation. For linear bandits, most of the papers that study lower bounds have been in context of lower bounds on regret [80, 242], typically under centralized setting. In the

distributed setting, to the best of our knowledge, the only work that discusses lower bounds especially in context of communication is by Amani *et al.* [11]. They derive a lower bound on the regret for the contextual linear bandit problem under communication constraints. However, in our work we study the lower bounds on communication costs under regret constraints, which is different from the problem considered in their work and requires significantly different analysis techniques.

### Explore-then-Commit

The algorithms proposed in [242, 332] are based on explore-then-commit type of approach which shares some features with our proposed algorithm, PLS. While there is a high-level similarity in the general approach of explore-then-commit, the structure of the norm estimation followed by refinement in PLS differentiates our work. This structure leads to adaptivity to the unknown  $\|\theta^*\|$  and the same regret order even when  $\|\theta^*\|$  is arbitrarily small. Differing from the self-adaptivity in PLS, the algorithm in [242] resorts to prior knowledge on the distribution of  $\theta^*$ , i.e., a Bayesian setting.

### Distributed Stochastic Optimization

There is an extensive literature on distributed stochastic optimization [89, 207, 181, 191, 199, 60, 96, 160, 161, 222, 230, 329, 328, 335, 315, 194, 82, 342]. Representative algorithms include Local-SGD<sup>1</sup> (also known as FedAvg) and Minibatch-SGD. In recent times, distributed stochastic optimization has received increased

---

<sup>1</sup>We collectively refer to all algorithms are based on the concept of Local-SGD, including the ones that employ momentum/variance reduction, to belong to this representative family.

interest in applications of Federated Learning (FL) [207]. FL aims to collaboratively learn a model by leveraging the data available at all the agents with a focus on ensuring privacy of the data for the participating agents. Developing communication efficient FL algorithms is also an active area of research.

It has been well-established that the algorithms in the aforementioned studies achieve order-optimal simple regret of  $O(1/MT)$  over a learning horizon of length  $T$  by using a weighted sum of all the iterates with a set of specifically designed weights based on the problem parameters. However, these results on simple regret do not necessarily imply sublinear, let alone order-optimal, guarantees on the cumulative regret performance of these algorithms. On the other hand, our proposed algorithm for distributed stochastic optimization, CEAL, achieves order-optimal cumulative regret, which is a finer measure of performance as compared to simple regret. Moreover, it can be shown that, using the convexity of the underlying function, the order-optimal cumulative regret of CEAL also implies an order-optimal (upto logarithmic factors) simple regret, resulting in significantly stronger performance guarantees over existing results. Furthermore, these guarantees are achieved by CEAL without the knowledge of specific problem parameters.

In terms of communication cost, there is a large body of work developing communication efficient algorithms by either reducing communication frequency [280, 164, 121, 334, 277, 335, 122, 209] or reducing message size [169, 31, 26, 281, 77, 283, 3, 232, 186, 345, 138, 149, 202, 321] by employing techniques like quantization and sparsification. See [284, 226] and [344] for a detailed survey of such approaches. However, as mentioned earlier, most of these works treat the two components of communication cost separately and hence focus on reducing

only one of these two components. Specifically, for Minibatch-SGD, the existing studies focus only on reducing communication frequency while allowing for high-precision message exchange. Our work deviates from existing works in its attempt to consider a holistic characterization of the communication cost.

## 5.2 Problem Formulation

Consider a system of  $M$  distributed agents indexed by  $\{1, 2, \dots, M\}$ . The agents face a common stochastic linear bandit model characterized by an unknown mean reward vector  $\theta^* \in \mathbb{R}^d$ . Specifically, each agent  $j \in \{1, 2, \dots, M\}$  has access to an action set  $\mathcal{A} = \{a \in \mathbb{R}^d : \|a\|_2 \leq 1\}$  and chooses to play an action  $a_t^j \in \mathcal{A}$  at every time instant  $t$  during a time horizon of  $T$  instants. When an action  $a_t^j \in \mathcal{A}$  is played by agent  $j$  at time  $t$ , it receives a reward

$$y_t^j = \langle \theta^*, a_t^j \rangle + \eta_t^j,$$

where  $\eta_t^j$  is zero-mean noise that is i.i.d. across time instants and across the agents and satisfies  $\log(\mathbb{E}[\exp(\lambda\eta_t^j)]) \leq \lambda^2\sigma^2/2$  for all  $\lambda \in \mathbb{R}$ , i.e., the noise is  $\sigma^2$ -sub-Gaussian. WLOG, we assume  $\|\theta^*\|_2 \leq 1$ . It is straightforward to extend it to the case  $\|\theta^*\|_2 \leq B$ , where  $B$  is a known constant, by appropriately scaling the obtained rewards. We point out that the unit ball assumption on the action space is adopted to facilitate a simpler exposition with greater focus on the impact of communication constraints. The algorithm can be easily extended to general action spaces (see Section 5.6.2).

Information exchange across the agents goes through a central server. Both the uplink channel (from the agents to the server) and downlink channel (from the server to the agents) have a finite capacity of  $R$  bits per channel use, which

limits the message size in each information exchange. This model quantifies the cumulative communication cost to the bit level, and represents a more challenging problem than those considered in the literature where communication channel between the server and the agent is assumed to have infinite capacity in at least one direction, if not both.

The design objective is a distributed learning policy consisting of (i) a decision strategy that governs the selection of actions  $\{a_t^j\}$  of each agent  $j$  at each time  $t$  and (ii) a communication strategy that determines when to communicate what and how to send it over the channel via quantization and encoding. The performance of a learning policy is measured in terms of the overall cumulative regret  $R(T)$  and the cumulative communication cost  $C(T)$  incurred by the policy. The overall cumulative regret is given by

$$R(T) = \sum_{j=1}^M \sum_{t=1}^T \left[ \max_{a \in \mathcal{A}} \langle \theta^*, a \rangle - \langle \theta^*, a_t^j \rangle \right]. \quad (5.1)$$

The communication cost  $C(T)$  is measured using  $C_u(T)$  and  $C_d(T)$ , the number of bits transmitted on the uplink channel (i.e., by any agent to the server) and that on the downlink channel (i.e., the average number of bits transmitted by the server to an agent), respectively.

The learning and communication efficiency of a learning policy is measured against the benchmarks. In particular, the cumulative regret is lower bounded by  $\Omega(d \sqrt{MT})$ , which is the optimal regret order in a centralized setting with total  $MT$  reward observations centrally available for learning. In Section 5.5, we establish an information-theoretic lower bound on the communication cost required for achieving a sublinear regret order.

## 5.3 Progressive Learning and Sharing

In this section, we present the Progressive Learning and Sharing (PLS) algorithm. We start in Section 5.3.1 with the basic structure of the algorithm followed by a detailed implementation in Section 5.3.2.

### 5.3.1 The Basic Structure of PLS

In PLS, learning and information sharing progress along the binary expansion of  $\theta^*$  in each dimension, starting from the most significant bit. Below we present separately the information sharing and learning components of PLS.

#### Progressive Information Sharing

In PLS, the unknown reward vector  $\theta^*$  is learnt with increasing accuracy, one bit at a time, as the algorithm progresses. Once the next bit in the binary expansion<sup>2</sup> of  $\theta^*$  in each of the  $d$  coordinates is learnt with sufficient accuracy, the agents transmit their estimates of this bit to the central server, which aggregates the estimates and broadcast the aggregated estimate of the new bit to the agents for subsequent exploitation and further exploration of  $\theta^*$ .

This progressive sharing mechanism can be seamlessly integrated with regret minimization to achieve both minimum regret order and minimum channel usage. Specifically, since only 1 bit of information is shared per coordinate in each information exchange, it suffices to send  $d$  bits in each transmission,

---

<sup>2</sup>While the algorithm learns an one bit at a time, it does not necessarily imply that the bit sequence learnt corresponds to the binary expansion.

achieving the benefit of having small messages. Furthermore, one can note that any reward-maximizing action taken based on an estimate  $\hat{\theta}$  incurs a regret proportional to the estimation error  $\|\hat{\theta} - \theta^*\|_2$ . Consequently, an estimation error of  $O(1/\sqrt{T})$  is sufficient to ensure an order-optimal regret, implying that it is sufficient to estimate each coordinate of  $\theta^*$  up to an accuracy of  $O(1/\sqrt{T})$ . Since sending the first  $r$  bits of the binary representation ensures an error of no more than  $2^{-r}$ , transmitting the first  $O(\log T)$  bits of the binary representation is sufficient to achieve the required accuracy. As a result, infrequent communication with a total of  $O(\log T)$  rounds can transmit all relevant information about  $\theta^*$ .

## Progressive Collaborative Learning

The progressive learning component of PLS is carried out in two stages: an initial stage for estimating the norm of  $\theta^*$  followed by a refinement stage with interleaving exploration and exploitation.

In the initial norm estimation stage, the goal is to estimate, within a multiplicative factor, the norm  $\|\theta^*\|_2$  of the underlying mean reward. This procedure is purely exploratory in nature. The collaborative exploration across agents is carried out in epochs with exponentially growing epoch lengths. At the end of each epoch, a threshold-based termination test is employed to determine whether the required estimation accuracy has been reached, which terminates the norm estimation stage. Information exchange occurs at the end of each epoch, and the exponentially growing epoch length ensures a low communication frequency.

The norm estimation stage serves multiple purposes. First, it allows PLS to be adaptive to the norm of  $\theta^*$  through the threshold-based termination rule.

Specifically, it provides sufficient initial exploration with sufficiency automatically adapted to  $\|\theta^*\|_2$  to ensure the estimation error is small enough for subsequent exploitation in the refinement stage. Second, this initial norm estimate sets the dynamic range of subsequent estimates of  $\theta^*$  to be used in the differential quantization for subsequent information sharing. Third, the estimate of  $\|\theta^*\|_2$  is also used to control the length of the exploitation epoch in the refinement stage to balance the exploration-exploitation trade-off.

In the refinement stage, the estimate of  $\theta^*$  obtained in the norm estimation stage is further refined. Similar to the norm estimation stage, the refinement stage also proceeds in epochs. The difference is that each epoch in the refinement stage consists of an exploration sub-epoch followed by an exploitation one. The exploration sub-epochs are for continual learning of  $\theta^*$ , one bit in each sub-epoch. The exploitation sub-epochs are to maximize rewards at each agent by playing the best action based on the current estimate of  $\theta^*$ . The lengths of the exploration and exploitation sub-epochs are both growing exponentially, but at different rates to carefully balance the exploration-exploitation trade-off. Information sharing is carried out only at the end of each exploration sub-epoch for the newly learned bit of  $\theta^*$ . The refinement stage with its interleaved exploration and exploitation continues until the end of the time horizon.

### 5.3.2 Detailed Description of PLS

In this section, we dive into the details of PLS.

## Progressive Collaborative Learning

**Norm Estimation Stage:** This stage proceeds in purely exploratory epochs. During an epoch  $k$ , each agent plays each unit vector in an orthonormal basis<sup>3</sup> of  $\mathbb{R}^d$  for  $s_k$  times. Each agent  $j$  computes the sample mean of the observed rewards for each basis vector to obtain an estimate  $\hat{\theta}_k^{(j)}$  of the underlying vector  $\theta^*$ . This estimate  $\hat{\theta}_k^{(j)}$  is clipped to within a radius of  $R_k + B_k$ , quantized using a stochastic quantizer with resolution  $\alpha_k$  and sent to the server by the agent. The process is repeated in every epoch until the agents receive a message from the server to terminate. We defer the details of the clipping and quantization steps to the Section 5.3.2 that describes the communication strategy. All policy parameters are specified at the end of the section.

At the server, upon receiving the estimates from the agents, the server averages them to obtain a combined estimate  $\hat{\theta}_k^{(\text{SERV})}$ . The server compares the norm of this estimate to a threshold  $4\tau_k$ . If the norm exceeds the threshold, the server sends a message to the agents to terminate the norm estimation stage. Otherwise, the server and the agents proceed into the next epoch. The value of  $\tau_k$  is chosen to be an upper bound on the estimation error of  $\theta^*$  at the end of the  $k^{\text{th}}$  epoch, allowing PLS to estimate  $\|\theta^*\|_2$  within a multiplicative factor at the end of the norm estimation stage.

The pseudo code for the norm estimation stage is given in Algorithms 6 and 7.

---

<sup>3</sup>The basis is chosen *a priori* and known to all the agents and the server. It can be any orthonormal basis of  $\mathbb{R}^d$ .

---

**Algorithm 6** Norm Estimation: Agent  $j \in \{1, 2, \dots, M\}$ 

---

```
1: Set  $k \leftarrow 1$ 
2: while True do
3:   Play each basis vector  $s_k$  times and compute the sample mean  $\hat{\theta}_k^{(j)}$ 
4:    $\tilde{\theta}_k^{(j)} \leftarrow \text{CLIP}(\hat{\theta}_k^{(j)}, R_k + B_k)$ 
5:    $Q(\tilde{\theta}_k^{(j)}) \leftarrow \text{STOQUANT}(\tilde{\theta}_k^{(j)}, \alpha_k, R_k + B_k)$ 
6:   Send  $Q(\tilde{\theta}_k^{(j)})$  to the server
7:   if received terminate from server then
8:     break
9:   else
10:     $k \leftarrow k + 1$ 
11:   end if
12: end while
```

---

---

**Algorithm 7** Norm Estimation: The Server

---

```
1: Set  $k \leftarrow 1$ 
2: while True do
3:   Compute  $\hat{\theta}_k^{(\text{SERV})} = \frac{1}{M} \sum_{j=1}^M Q(\tilde{\theta}_k^{(j)})$ 
4:   if  $\tau_k \leq \frac{1}{4} \|\hat{\theta}_k^{(\text{SERV})}\|$  then
5:     Server sends terminate to all agents
6:     break
7:   else
8:      $k \leftarrow k + 1$ 
9:   end if
10: end while
```

---

**Refinement Stage:** This stage also proceeds in epochs, starting with the epoch index at which Norm Estimation stage terminated. Each epoch  $k$  during Refinement begins with an exploration sub-epoch where, similar to Norm Estimation, each agent obtains their estimate of  $\theta^*$ ,  $\hat{\theta}_k^{(j)}$ , by playing each of the basis vectors  $s_k$  times. At the end of the sub-epoch, the agents share the next bit learnt during this time by transmitting  $\hat{\theta}_k^{(j)} - \bar{\theta}_{k-1}$  to the server after appropriate clipping and quantization. Here  $\bar{\theta}_{k-1}$  denotes the current estimate of  $\theta^*$  available to the agents after  $k - 1$  epochs. This differential quantization allows the agents to share only the “new bit” learnt during the exploration sub-epoch. As a re-

sponse, the agents receive  $Q(\hat{\theta}_k^{(\text{SERV})})$ , a quantized version of the update, from the server which is used to refine their estimate of  $\theta^*$  to  $\bar{\theta}_k = \bar{\theta}_{k-1} + Q(\hat{\theta}_k^{(\text{SERV})})$ . The exploitation sub-epoch follows the communication round where all agents play the unit vector along  $\bar{\theta}_k$  throughout the  $t_k$  time steps of the sub-epoch. The refinement stage continues by proceeding into the next epoch and repeating the process until the end of the time horizon.

The steps at the server in this stage are similar to those in the norm estimation stage. In particular, the server collects estimates from the agents at the end of the exploration sub-epoch, computes the mean  $\hat{\theta}_k^{(\text{SERV})}$  and then broadcasts the differential update  $\hat{\theta}_k^{(\text{SERV})} - \bar{\theta}_{k-1}$  after passing it through a deterministic quantizer with resolution  $\beta_k$ . A pseudo code for the refinement stage is given in Algorithms 8 and 9.

---

**Algorithm 8** Refinement: Agent  $j \in \{1, 2, \dots, M\}$ 


---

```

1: Input: The epoch index at the end of Norm Estimation stored as  $k_0$ ,  $\bar{\theta}_{k_0-1} \leftarrow 0$ ,  $k \leftarrow k_0$ 
2: while budget is not exhausted do
3:   Play each basis vector  $s_k$  times and compute the sample mean  $\hat{\theta}_k^{(j)}$ 
4:    $\tilde{\theta}_k^{(j)} \leftarrow \text{CLIP}(\hat{\theta}_k^{(j)} - \bar{\theta}_{k-1}, R_k + B_k)$ 
5:    $Q(\tilde{\theta}_k^{(j)}) \leftarrow \text{STOQUANT}(\tilde{\theta}_k^{(j)}, \alpha_k, R_k + B_k)$ 
6:   Send  $Q(\tilde{\theta}_k^{(j)})$  to the server
7:   Receive  $Q(\hat{\theta}_k^{(\text{SERV})})$  from the server
8:    $\bar{\theta}_k \leftarrow \bar{\theta}_{k-1} + Q(\hat{\theta}_k^{(\text{SERV})})$ 
9:   if  $k = k_0$  then
10:    Set  $\mu_0 \leftarrow \|\bar{\theta}_k\|_2$ 
11:   end if
12:   Play the action  $a = \bar{\theta}_k / \|\bar{\theta}_k\|$  for the next  $t_k$  rounds.
13:    $k \leftarrow k + 1$ 
14: end while

```

---

**Setting Policy Parameters:** We now specify the values of parameters used in PLS. For an epoch  $k$ , the length of the exploration (sub-)epoch,  $s_k$ , is set to

---

**Algorithm 9** Refinement: The Server

---

```

1: Input: The epoch index at the end of Norm Estimation stored as  $k_0$ ,  $\bar{\theta}_{k_0-1} \leftarrow 0$ ,  $k \leftarrow k_0$ 
2: while time horizon  $T$  is not reached do
3:   Receive  $Q(\tilde{\theta}_k^{(j)})$  from all the agents
4:   Compute  $\hat{\theta}_k^{(\text{SERV})} = \bar{\theta}_{k-1} + \frac{1}{M} \sum_{j=1}^M Q(\tilde{\theta}_k^{(j)})$ 
5:    $Q(\hat{\theta}_k^{(\text{SERV})}) \leftarrow \text{DETQUANT}(\hat{\theta}_k^{(\text{SERV})} - \bar{\theta}_{k-1}, \beta_k, B_k + \tau_k)$  and broadcasts it to all agents
6:    $k \leftarrow k + 1$ 
7: end while

```

---

$\lceil 40\sigma^2 d \log(8MK/\delta)4^k \rceil$  and that of the exploitation one is set to  $t_k := \lceil Ms_k^2\mu_0^2 \rceil$ . In the above definitions,  $K$  denotes the maximum possible number of epochs in the algorithm and is defined as  $K := \max\{k \in \mathbb{N} : 40\sigma^2 d \log(8Mk/\delta)(4^k - 4) \leq T\} = O(\log T)$ . In the definition of  $t_k$ ,  $\mu_0$  is the estimate of  $\|\theta^*\|_2$  obtained at the end of the norm estimation stage. The exponential lengths of the epoch designed for the bit by bit progressive learning are evident from the above choices. This choice of the lengths also allows PLS to address the exploration-exploitation trade-off by balancing the regret incurred during the exploration and exploitation sub-epochs.

The threshold  $\tau_k$  and sequence  $R_k$  are set based upon high probability bounds on  $\|\hat{\theta}_k^{(\text{SERV})} - \theta^*\|_2$  and  $\|\hat{\theta}_k^{(j)} - \theta^*\|_2$  respectively that simultaneously hold for all agents. In particular,  $\tau_k$  is set to  $3 \cdot 2^{-(k+1)} / \sqrt{M}$  and  $R_k := 2^{-k}$ . Notice that this choice of  $R_k$  echoes the progressive learning feature, allowing the agents to learn  $\theta^*$ , one bit at a time. The sequence  $B_k$  bounds the error  $\|\bar{\theta}_{k-1} - \theta^*\|_2$  and is set to  $5\tau_k$  for  $k \geq 2$  with  $B_1 = 1$ . Lastly, the resolution parameter sequences  $\alpha_k$  and  $\beta_k$  are defined as  $\alpha_k := \alpha_0 \sigma \sqrt{d} / \sqrt{s_k}$  and  $\beta_k = \beta_0 \tau_k$  for some numerical constants  $\alpha_0, \beta_0 < 1$ . The constants  $\alpha_0$  and  $\beta_0$  control the message size associated with uplink and downlink communication respectively.

## Progressive Information Sharing

The communication protocol of PLS consists of two steps : clipping and quantizing the vector to be sent to reduce the size of message being transmitted and encoding the quantized value to send it over the communication channel.

**Clipping and Quantization:** This step employs well-known sub-routines described below to map a vector to a low resolution, quantized version of itself.

- $\text{CLIP}(x, r)$  is a simple routine that takes input a vector  $x$  and clips it within a  $\ell_2$  ball of radius  $r$ . Mathematically, the routine returns the value  $x \cdot \min\{1, r/\|x\|\}$ .
- $\text{STOQUANT}(y, \varepsilon, r)$  returns the quantized version of a scalar  $y$  using the popular approach of stochastic quantization. Specifically, the interval  $[-r, r]$  is divided into  $l_\varepsilon = \lceil 2r/\varepsilon \rceil$  intervals of equal length and indexed from 1 to  $l_\varepsilon$ . The value  $y$  is quantized to one of the end points of the intervals to which it belongs in a randomized manner with the probability inversely proportional to the distance from  $y$ . In particular, the stochastic quantizer outputs  $Q_s(y)$  given by

$$Q_s(y) = \begin{cases} b_{l-1} & \text{w.p. } b_l - y, \\ b_l & \text{otherwise.} \end{cases}$$

In the above expression,  $b_m = r\left(\frac{2m}{l_\varepsilon} - 1\right)$  for  $m = 1, 2, \dots, l_\varepsilon$  and  $l = \{m : b_{m-1} \leq y < b_m\}$ . With a slight abuse of notation, we also use  $\text{STOQUANT}(x, \varepsilon', r)$  to denote stochastic quantization of a vector  $x$ . In the case of a vector, all the coordinates are quantized as mentioned above with  $\varepsilon = \varepsilon'/\sqrt{d}$ .

- $\text{DETQUANT}(y, \varepsilon, r)$  is also a quantization routine similar to  $\text{STOQUANT}(y, \varepsilon, r)$  with the only difference that the output of this routine is deterministic. Specifically, the deterministic quantizer outputs

$$Q_d(y) = \begin{cases} b_{l-1} & \text{if } |b_l - y| > |b_{l-1} - y|, \\ b_l & \text{otherwise.} \end{cases}$$

Once again we overload the definition with a vector analogue  $\text{DETQUANT}(x, \varepsilon', r)$  where each coordinate is deterministically quantized with  $\varepsilon = \varepsilon' / \sqrt{d}$ .

The two different quantization schemes used in this step serve their own, different purposes in algorithm. PLS employs stochastic quantization for uplink communication which allows it to exploit the concentration properties of the zero mean noise added by the quantization. Consequently, it enables to completely leverage the access to observations from  $M$  different agents to achieve the speed up proportional to the number of agents. A deterministic quantization scheme in its place would have resulted in accumulation of errors and consequently prevented the speedup with the number of agents. On the other hand, the deterministic quantization used for downlink transmission ensures that all the agents have the same estimate of  $\theta^*$  and consequently the same value of  $\mu_0$ , the estimate of  $\|\theta^*\|_2$ . Since  $\mu_0$  governs the length of the exploitation sub-epoch, a common value shared between the agents helps maintain the synchronization across different epochs and agents.

**Encoding:** As described above, both the quantization routines used in PLS when run with accuracy parameter  $\varepsilon$ , quantize each coordinate axis into  $l_\varepsilon$  intervals resulting in  $l_\varepsilon + 1$  possible values for the quantized version of each co-

ordinate. The encoding step maps these  $(l_\epsilon + 1)^d$  possible values of quantized vectors to different messages that can be sent over the communication channel. We use a common encoding strategy for the uplink and downlink channels.

PLS sends each coordinate of the quantized version, one by one, using the variable-length encoding strategy, unary coding [74]. In particular, each coordinate is represented by a header followed by a sequence of 1's whose length is equal to the absolute value of the coordinate. The header is 3 bits long where the first and the third bit are both 0 and the second bit represents the sign of the value, where 0 & 1 imply negative and positive values respectively. As an example, in this encoding scheme, the numbers  $-3$  and  $5$  are represented as  $000111$  and  $01011111$  respectively.

## 5.4 Performance Analysis

In this section, we show that PLS achieves the order-optimal regret of  $O(d \sqrt{MT})$  up to logarithmic factors with a communication cost that matches the order of information-theoretic lower bound established in Section 5.5.

### 5.4.1 Regret Analysis

The following theorem characterizes the regret of PLS.

**Theorem 5.4.1.** *Consider the distributed stochastic linear bandit setting described in Section 5.2. If PLS is run with parameters as described in Section 5.3.2 for  $T$  time*

instants, then the following relation holds with probability at least  $1 - \delta$ ,

$$R_{PLS}(T) \leq Cd\sqrt{MT} \log\left(\frac{8MK}{\delta}\right) \log(9\sqrt{MT}) \\ + O(\log T),$$

for some constant  $C > 0$ , independent of  $d, M, \delta$  and  $T$ .

Theorem 5.4.1 establishes that the regret incurred by PLS is  $\tilde{O}(d\sqrt{MT})$  which matches the lower bound for a centralized setting with  $MT$  queries up to logarithmic factors. This implies that PLS explores the achievability of communication-learning trade-off at the frontier of optimal learning performance.

We here provide a sketch of the proof for Theorem 5.4.1. We begin the proof by decomposing the regret incurred by PLS as follows :  $R_{PLS}(T) = R_{NE}(T) + R_{REF}(T)$ , where  $R_{NE}(T)$  and  $R_{REF}(T)$  denote the regret incurred during the norm estimation and the refinement stages respectively. The bound on regret incurred by PLS is obtained by separately bounding each of above the two terms in the decomposition. The following lemma provides a bound on the regret incurred during the norm estimation stage.

**Lemma 5.4.2.** *Consider the Norm Estimation stage described in Alg. 6 and 7. If it is run for at most  $T$  time instants in a distributed setup with  $M$  agents and  $\|\theta^*\| \leq 1$ , then the regret incurred during this stage satisfies the following relation with probability at least  $1 - \delta$ ,*

$$R_{NE}(T) \leq Cd(1 + \sigma^2)\sqrt{MT} \log(1/\delta') \log_2(9\sqrt{MT}) \\ + 2d \log_2(9\sqrt{MT}),$$

where  $\delta' = \delta/8MK$  and  $C > 0$  is a constant independent of  $d, M, \delta$  and  $T$ .

Since the Norm Estimation stage is based on pure exploration, we upper bound  $R_{\text{NE}}(T)$  by  $2\|\theta^*\|_2$  times the duration of the norm estimation stage, where  $2\|\theta^*\|_2$  corresponds to the trivial bound on the instantaneous regret. The central step in the proof of the above lemma is bounding the duration of the norm estimation stage. For this part, we first establish a  $O(1/\|\theta^*\|_2^2)$  bound on the duration based on our threshold test which determines when the stage terminates. However, the fixed length of the time horizon dictates a hard upper bound of  $T$  on the duration of this stage. The proof is completed by taking a minimum of these two bounds to bound the duration of the norm estimation stage followed optimizing over  $\|\theta^*\|_2$  to obtain the tightest bound.

To bound  $R_{\text{REF}}(T)$ , we separately bound the regret incurred during the exploration and exploitation sub-epochs. The regret incurred during the exploration sub-epoch of the  $k^{\text{th}}$  epoch is bounded by  $2\|\theta^*\|_2$  times  $ds_k$ , the duration of the exploration sub-epoch, similar to the proof of Lemma 5.4.2. The regret incurred during the exploitation sub-epoch is bounded using the following lemma, which is similar to Lemma 3.6 in [242].

**Lemma 5.4.3.** *If  $\hat{\theta}$  is an estimate of a vector  $\theta$  such that  $\|\hat{\theta} - \theta\|_2 \leq \tau \leq \|\theta\|_2$ , then instantaneous regret incurred by an algorithm that plays the action  $a = \hat{\theta}/\|\hat{\theta}\|_2$  on a stochastic linear bandit instance with true underlying vector  $\theta$  can be bounded by  $\tau^2/\|\theta\|$ .*

Lemma 5.4.3 implies that even when the estimation error is  $\nu\|\theta^*\|_2$ , for  $\nu \in [0, 1]$  the regret incurred by the algorithm scales as  $\nu^2\|\theta^*\|_2$ . This is a crucial fact that helps balance the exploration-exploitation trade-off. In particular, the lengths of the exploration and exploitation epochs in PLS are designed based on the result of this lemma. The estimation error at the end of explo-

ration sub-epoch in epoch  $k$  satisfies  $O(s_k^{-1/2})$  while the regret incurred during the sub-epoch satisfies  $O(s_k \|\theta^*\|_2)$ . Based on the above lemma, the regret incurred during the corresponding exploitation sub-epoch of length  $t_k = O(s_k^2 \|\theta^*\|_2^2)$  is  $O(\frac{1}{s_k \|\theta^*\|_2} \cdot t_k) = O(s_k \|\theta^*\|_2)$  matching the regret incurred during the exploration phase. This is the fundamental mechanism that provides the necessary balance between exploration and exploitation in PLS allowing it to achieve optimal-order regret. Moreover, this also provides additional insight into the novel choice of the length exploitation epoch based on the norm of  $\theta^*$  in PLS.

The final bound on  $R_{\text{REF}}(T)$  is obtained by noting each epoch is  $O(s_k^2 + s_k) = O(16^k)$  steps long implying a total of  $O(\log T)$  epochs. The detailed proofs of all the Lemmas and the Theorem can be found in Appendix D.1.

#### 5.4.2 Communication Cost

The communication cost incurred by PLS is characterized in the following theorem.

**Theorem 5.4.4.** *Consider the distributed stochastic linear bandit setting described in Section 5.2. If PLS is run with parameters as described in Section 5.3 for a time horizon of  $T$ , then the uplink and the downlink communication costs (in bits) incurred by PLS, i.e.  $C_u(T)$  and  $C_d(T)$ , satisfy  $O\left(\frac{d}{\alpha_0} \log T\right)$  and  $O\left(\frac{d}{\beta_0} (\log M + \log T)\right)$  respectively.*

Thus, Theorem 5.4.4 in conjunction with Theorem 5.4.1 shows that PLS incurs the order-optimal regret while simultaneously achieving the order-optimal communication cost matching the lower bound in Section 5.5. Hence, PLS further reduces the communication cost as compared to communication-efficient

algorithms proposed in [320, 143, 11] while maintaining the optimal regret performance. The proof of the above theorem revolves around the following lemma.

**Lemma 5.4.5.** *Consider the communication scheme of PLS outlined in Section 5.3.2. If PLS is run with parameters as described in Section 5.3.2, then any message exchanged between the server and an agent during PLS is at most  $O(d)$  bits.*

The bound on uplink communication cost  $C_u(T)$  immediately follows by noting that there are at most  $K = O(\log T)$  epochs, or equivalently, communication rounds in PLS. The additional  $O(d \log M)$  term in  $C_d(T)$  is incurred when the server sends the initial estimate of  $\theta^*$  to all the agents. The details of the proof along with proof of Lemma 5.4.5 are provided in the supplementary material.

*Remark 5.4.6.* Based on Lemma 5.4.5, it can immediately concluded that a capacity of  $R = O(d)$  bits for both the uplink and the downlink channel suffices for PLS to achieve order-optimal regret performance. Some other studies like Suresh *et al.* [282] and Mitra *et al.* [212] have also proposed encoding schemes which result in message sizes of  $O(d)$  bits. Suresh *et al.* [282] proposed an encoding scheme for distributed mean estimation using the well-known variable length encoding schemes such as Huffman and Arithmetic encoding. It first constructs a histogram over the different  $l_\epsilon + 1$  values in the quantized version of a vector followed by a Huffman tree based on that histogram. During each communication round, the sender first sends the corresponding Huffman tree followed by the message encoded using that. Compared to such variable-length schemes, our proposed scheme is easier to implement, both in terms of memory and computation, and also avoids overheads like sending the Huffman tree. The IC-Lin-UCB algorithm proposed in [212] uses an encoding scheme based

on constructing  $\varepsilon$ -cover of hyperspheres in  $\mathbb{R}^d$  at every time instant. The computational cost of constructing such covering sets grows exponentially with the dimension, rendering the approach infeasible even for problems of moderate dimensionality. On the other hand, the computational cost of the encoding scheme in PLS grows linearly with dimension, making PLS an attractive option even for high dimensional problems.

*Remark 5.4.7.* The  $O(d \log M)$  term in the downlink communication cost can be interpreted as the cost incurred by the server to facilitate information exchange since the data is distributed across  $M$  agents. In other words, the server provides each agent with the additional information learnt from the other agents through these  $O(d \log M)$  bits, leading to *collaboration* among them. In absence of transmission of these bits, the problem would reduce to  $M$  independent agents trying to learn a linear bandit model and incurring an overall regret of  $O(dM \sqrt{T})$  that grows linearly with  $M$ . Thus, it can also be interpreted as the cost associated with having a sublinear regret with respect to the number of agents  $M$ . Similarly,  $O(d \log T)$  term corresponds to the cost associated with having a sublinear regret with respect to the time horizon.

## 5.5 Lower Bound on Communication Cost

In this section, we explore the converse result for the communication-learning trade-off. In particular, the following theorem establishes information theoretic lower bounds in terms of actual number of bits that need to be transmitted by the clients and the server over the channel for any distributed algorithm to achieve sublinear regret.

**Theorem 5.5.1.** *Consider the distributed linear bandit instance with  $M$  agents described in Section 5.2. Any distributed algorithm that incurs an overall cumulative regret that is order optimal in both  $T$  and  $M$  with probability at least  $2/3$  needs to transmit at least  $\Omega(d \log(MT))$  bits of information over the downlink channel and  $\Omega(d \log T)$  bits of information over the uplink channel.*

The lower bounds established in the above theorem match the achievability results for PLS shown in the previous section. We would like to emphasize that PLS is the first algorithm for distributed linear bandits for which communication cost incurred matches the order of information theoretic lower bounds in terms of actual number of bits transmitted over the channel. Additionally, PLS simultaneously attains order-optimal regret guarantees. Thereby, the above theorem along with the performance guarantees of PLS provides a holistic view of the communication-learning efficiency trade-off in distributed linear bandits. The proof of the theorem follows from an application of Fano's inequality along with bounds on metric entropy. A detailed proof of the above theorem is can be found in Appendix D.2.

## 5.6 Extensions

### 5.6.1 Leveraging Sparsity

We now consider a variant of the original problem where the underlying reward vector  $\theta^*$  is known to satisfy an additional sparsity constraint. In particular, it is known that the number of non-zero elements in  $\theta^*$  are no more than  $s \ll d$ , i.e.,  $\|\theta^*\|_0 \leq s$ . For this setup, we propose Sparse-PLS, a variant of PLS, to leverage

the sparsity of  $\theta^*$  to further reduce the communication cost.

Sparse-PLS makes two modifications to the original PLS algorithm to leverage the sparsity of  $\theta^*$ . The first modification is made to the set of actions played during an exploration epoch. Specifically, Sparse-PLS replaces the orthonormal basis of  $R^d$  with a set of actions,  $\mathcal{B}_s$ , that spans only a subspace of  $R^d$ .  $\mathcal{B}_s$  consists of  $m = O(s \log(d/\delta)) \ll d$  vectors drawn independently from the set  $\{-1/\sqrt{d}, +1/\sqrt{d}\}^d$ . It is ensured that all the agents use the same random set  $\mathcal{B}_s$  via a common random seed. This modification offers a two-fold advantage. First, it helps reduce the message size required for uplink communication as it is sufficient to transmit these noisy projections in  $\mathbb{R}^m$ , where  $m \ll d$ , in order to recover the original sparse vector  $\theta^*$  [55, 53, 54]. Second, since the regret incurred in PLS is proportional to length of the exploration epochs, this modification allows Sparse-PLS to replace a factor of the actual dimension of the vector with the level of sparsity in the regret bounds, making it smaller (See Theorem 5.6.1). The second modification is made to the process to estimate  $\theta^*$  at the server. Sparse-PLS employs the LASSO estimator [289, 35] at the server to obtain a sparse estimate of  $\theta^*$  from the noisy projections sent by the agents. Pseudo-codes for Sparse-PLS are provided in Algorithms 10, 11, 12 and 13.

---

**Algorithm 10** Sparse-PLS Norm Estimation: Agent  $j \in \{1, 2, \dots, M\}$ 


---

```

1: Input: The set of actions  $\mathcal{B}_s$ 
2: Set  $k \leftarrow 1$ 
3: while True do
4:   Play each vector in  $\mathcal{B}_s$  for  $s_k$  times and compute the sample mean  $\hat{\theta}_k^{(j)}$ 
5:    $\tilde{\theta}_k^{(j)} \leftarrow \text{CLIP}(\hat{\theta}_k^{(j)}, R_k + B_k)$ 
6:    $Q(\tilde{\theta}_k^{(j)}) \leftarrow \text{STOQUANT}(\tilde{\theta}_k^{(j)}, \alpha_k, R_k + B_k)$ 
7:   Send  $Q(\tilde{\theta}_k^{(j)})$  to the server
8:   if received terminate from server then
9:     break
10:  else
11:     $k \leftarrow k + 1$ 
12:  end if
13: end while

```

---



---

**Algorithm 11** Sparse-PLS Norm Estimation: The Server

---

```

1: Input: The set of actions  $\mathcal{B}_s$ 
2: Set  $k \leftarrow 1$ 
3: while True do
4:   Compute  $\hat{\theta}_k^{(\text{SERV})} := \arg \min_{\theta} \frac{d}{m} \left\| \frac{1}{M} \sum_{j=1}^M Q(\tilde{\theta}_k^{(j)}) - X\theta \right\|_2^2 + \lambda_k \|\theta\|_1$ 
5:   if  $\tau_k \leq \frac{1}{4} \|\hat{\theta}_k^{(\text{SERV})}\|$  then
6:     Server sends terminate to all agents
7:     break
8:   else
9:      $k \leftarrow k + 1$ 
10:  end if
11: end while

```

---

In the pseudo codes for Sparse-PLS,  $X$  refers to the  $\mathbb{R}^{m \times d}$  matrix obtained by stacking the vectors in  $\mathcal{B}_s$  one under the other. The parameters for Sparse-PLS are set as  $m = 80(s \log(150d/s) + \log(4/\delta))$ ,  $s_k := \lceil 40\sigma^2 d \log(16MK/\delta) 4^k \rceil$ ,  $t_k := \lceil mMs_k^2/d \rceil$ ,  $R_k := 2^{-k} \sqrt{m/d}$ ,  $B_k = 7\tau_k \sqrt{m/d}$ ,  $\tau_k := 3 \cdot 2^{-(k+1)} / \sqrt{M}$ ,  $\alpha_k = \alpha_0 \sigma \sqrt{s/s_k}$ ,  $\beta_k = \beta_0 \tau_k$  and  $\lambda_k := 4\sigma \sqrt{\frac{3}{2ms_k}} (\sqrt{\log(2d)} + \sqrt{\log(4/\delta)})$ . The underlying philosophy behind the choice of PLS-specific parameters is similar to that of PLS. The additional parameters  $\lambda_k$  and  $m$  are chosen based on the results in recovery of sparse signals [35, 25].

---

**Algorithm 12** Sparse-PLS Refinement: Agent  $j \in \{1, 2, \dots, M\}$ 


---

- 1: **Input:** The epoch index at the end of Norm Estimation stored as  $k_0$ , The set of actions  $\mathcal{B}_s$
- 2:  $\bar{\theta}_{k_0-1} \leftarrow 0, k \leftarrow k_0$
- 3: **while** time horizon  $T$  is not reached **do**
- 4:   Play each vector in  $\mathcal{B}_s$  for  $s_k$  times and compute the sample mean  $\hat{\theta}_k^{(j)}$
- 5:    $\tilde{\theta}_k^{(j)} \leftarrow \text{CLIP}(\hat{\theta}_k^{(j)} - \bar{\theta}_{k-1}, R_k + B_k)$
- 6:    $Q(\tilde{\theta}_k^{(j)}) \leftarrow \text{STOQUANT}(\tilde{\theta}_k^{(j)}, \alpha_k, R_k + B_k)$
- 7:   Send  $Q(\tilde{\theta}_k^{(j)})$  to the server
- 8:   Receive  $Q(\hat{\theta}_k^{(\text{SERV})})$  from the server
- 9:    $\bar{\theta}_k \leftarrow \bar{\theta}_{k-1} + Q(\hat{\theta}_k^{(\text{SERV})})$
- 10:   **if**  $k = k_0$  **then**
- 11:     Set  $\mu_0 \leftarrow \|\bar{\theta}_k\|_2$
- 12:   **end if**
- 13:   Play the action  $a = \bar{\theta}_k / \|\bar{\theta}_k\|$  for the next  $t_k$  rounds.
- 14:    $k \leftarrow k + 1$
- 15: **end while**

---

**Algorithm 13** Sparse-PLS Refinement: The Server

---

- 1: **Input:** The epoch index at the end of Norm Estimation stored as  $k_0$ , The set of actions  $\mathcal{B}_s$
- 2:  $\bar{\theta}_{k_0-1} \leftarrow 0, k \leftarrow k_0$
- 3: **while** time horizon  $T$  is not reached **do**
- 4:   Receive  $Q(\tilde{\theta}_k^{(j)})$  from all the agents
- 5:   Compute  $\hat{\theta}_k^{(\text{SERV})} = \bar{\theta}_{k-1} + \arg \min_{\theta} \frac{d}{m} \left\| \frac{1}{M} \sum_{j=1}^M Q(\tilde{\theta}_k^{(j)}) - X(\theta - \bar{\theta}_{k-1}) \right\|_2^2 + \lambda_k \|\theta\|_1$
- 6:    $Q(\hat{\theta}_k^{(\text{SERV})}) \leftarrow \text{DETQUANT}(\hat{\theta}_k^{(\text{SERV})} - \bar{\theta}_{k-1}, \beta_k, B_k + \tau_k)$  and broadcasts it to all agents
- 7:    $k \leftarrow k + 1$
- 8: **end while**

---

The following theorem characterizes the performance of Sparse-PLS,in terms of both regret and communication cost.

**Theorem 5.6.1.** *Consider the distributed stochastic linear bandit setting described in Section 5.2 with an additional assumption of sparsity on the underlying mean reward, i.e.,  $\|\theta^*\|_0 \leq s$ . If Sparse-PLS is run for a time horizon of  $T$ , then the regret incurred by*

*Sparse-PLS* satisfies

$$R_{\text{Sparse-PLS}}(T) \leq C \sqrt{sdMT} \log(MK/\delta) \log(\sqrt{MT}),$$

with probability at least  $1 - \delta$  for some  $C > 0$ , independent of  $s, d, M$  and  $T$ . Moreover, the uplink and downlink communication costs,  $C_u(T)$  and  $C_d(T)$  are no more than  $O(s \log T)$  and  $O(d(\log M + \log T))$  bits respectively.

As it can be noted from the above theorem, Sparse-PLS replaces a factor of dimension  $d$  with the sparsity level  $s$ , or equivalently the effective dimension, in the regret bound matching the optimal-order regret bounds [2]. Furthermore, it also reduces  $C_u(T)$  from  $O(d \log T)$  to  $O(s \log T)$  demonstrating its ability to leverage the inherent sparsity to reduce communication and regret. We refer the reader to Appendix D.3.1 for a detailed proof.

### 5.6.2 Beyond the Unit Ball

The unit ball assumption on the action space can be relaxed to smooth action spaces without any change to the algorithm. We refer an action space  $\mathcal{A}$  to be smooth if for any  $\theta, \theta'$  in the unit ball,  $\|a(\theta) - a(\theta')\| \leq L\|\theta - \theta'\|$  for some  $L > 0$ , where  $a(\theta) = \arg \max_{v \in \mathcal{A}} \langle v, \theta \rangle$ . The smoothness condition allows one to directly use the estimate of  $\theta^*$  as a proxy for  $\theta^*$  for pure exploitation. In particular, for  $\hat{\theta}$  satisfying  $\|\theta^* - \hat{\theta}\| \leq \tau$ , the regret incurred by using  $\hat{\theta}$  as a proxy can be written as  $\langle a(\theta^*), \theta^* \rangle - \langle a(\hat{\theta}), \hat{\theta} \rangle \leq \|a(\theta^*) - a(\hat{\theta})\| \|\theta^* - \hat{\theta}\| \leq L\tau^2$ . This result is generalization of Lemma 5.4.3 to smooth action spaces. The rest of the analysis would follows as is, yielding the result.

The unit ball assumption on the action space can be completely done away

with to allow for general arbitrary actions by making two minor modifications to the PLS algorithm. Firstly, during the exploration sub-epoch, the orthogonal basis of  $\mathbb{R}^d$  is replaced by a  $\mathcal{B} \subseteq \mathcal{A}$ , where the set  $\mathcal{B}$  satisfies  $|\mathcal{B}| = d$ ,  $\text{span}(\mathcal{B}) = \mathbb{R}^d$  and that all the eigenvalues of  $\sum_{x \in \mathcal{B}} xx^\top$  are greater than some  $\lambda > 0$ . The existence of such a set  $\mathcal{B}$  is a very mild assumption on the action set. We also set the length of the exploration sub-epoch  $s_k = \lceil 40(\sigma/\lambda)^2 d \log(8MK/\delta) 4^k \rceil$  and modify the remaining parameters appropriately. Secondly, on account of the possible non-smoothness of the action set, pure exploitation using  $\hat{\theta}$  as a proxy may no longer yield optimal regret guarantees as the optimal action may change by a lot even by a small perturbation in  $\theta^*$ . The non-smoothness necessitates a small amount of continuous exploration in the “exploitation” phase to avoid sticking on one action, which we ensure using the uncertainty ellipsoid technique. In particular, for the  $r^{\text{th}}$  instant during the exploitation sub-epoch of the  $k^{\text{th}}$  epoch, take the action  $A_r = \arg \max_{a \in \mathcal{A}_k} \langle a, \check{\theta}_{r-1} \rangle + \Gamma a^\top C_{r-1}^{-1} a$ . In the above description,  $\mathcal{A}_k = \{a \in \mathcal{A} : \langle a, \bar{\theta}_k \rangle \geq \max_{a' \in \mathcal{A}} \langle a', \bar{\theta}_k \rangle - 2\tau_k\}$ ,  $C_r = C_{r-1} + A_r A_r^\top$ ,  $\check{\theta}_{r-1} = C_{r-1}^{-1} \sum_{l=1}^r A_l (y_l - \langle A_l, \bar{\theta}_k \rangle)$ , where  $A_l$  denotes the action taken at the  $l^{\text{th}}$  instant,  $y_l$  denotes the corresponding reward,  $C_0 = I$  (the identity matrix) and  $\Gamma > 0$  is an appropriately chosen constant.

The analysis for the modified version of PLS is very similar to that of the original one. Since  $\mathcal{B}$  also spans  $\mathbb{R}^d$ , the error bounds on  $\hat{\theta}^{(\text{SERV})}$  follow as is. For the regret analysis, the regret for the exploration sub-epochs remains unchanged. In the “exploitation” sub-epoch, we can use result from [242, 69] to conclude that the regret incurred during the sub-epoch is  $O(d \sqrt{t_k})$ , which upon substituting the value of  $t_k$  yields the same bound as in the current version, upto constants. The final bound on the regret then follows exactly as described in Appendix D.1.3.

## 5.7 From Linear Bandits to Convex Functions

In this section, we extend the insights obtained for the problem of distributed linear bandits to study the problem of first-order online stochastic convex optimization in a distributed setup. Under this setup,  $M$  clients aim to collaboratively minimize an unknown convex function  $f : \mathcal{X} \rightarrow \mathbb{R}$  by using noisy gradient estimates<sup>4</sup> at sequentially queried points in the domain,  $\mathcal{X} \subset \mathbb{R}^d$ , assumed to be convex, compact set. The function  $f$  is known to be  $\alpha$ -strongly convex and  $\beta$ -smooth.

One key performance metric for stochastic optimization is the learning efficiency. In existing studies, the offline measure of simple regret is commonly adopted as the measure for learning efficiency. Specifically, simple regret is defined as  $f(\hat{x}_T) - \min f$ , where  $\hat{x}_T$  is a point returned by the algorithm at the end of the learning horizon and corresponds to the best estimate of the minimizer learnt by the algorithm. Since simple regret only cares about sub-optimality gap of the learned minimizer, it does not reflect the performance of an algorithm in an online setting, where it is important to control the running sum of excessive loss in real time during the learning process. A more appropriate metric in online settings is cumulative regret, which provides a cumulative assessment of the learning throughout the algorithm as is often needed in online learning setups.

In contrast to existing studies, we adopt the online measure of cumulative regret for learning efficiency and a holistic measure for communication effi-

---

<sup>4</sup>If at time instant  $t$ , client  $m$  queries a point  $x_t^m$ , it observes a noisy observation of gradient at  $x_t^m$ , given by  $G(x_t^m) = \nabla f(x_t^m) + \xi_t^m$ .  $\{\xi_t^m\}_{m,t}$  are i.i.d. random vectors and correspond to the noise in the observations. They are known to be zero mean,  $\sigma^2$ -sub Gaussian random vectors, i.e., they satisfy  $\mathbb{E}[\exp(\lambda v^\top \xi_t^m)] \leq \exp(\lambda^2 \sigma^2 / 2d)$  for all  $\lambda \in \mathbb{R}$ ,  $t \in \{1, 2, \dots, T\}$ ,  $m \in \{1, 2, \dots, M\}$  and unit vectors  $v \in \mathbb{R}^d$ .

ciency. We develop a distributed online learning algorithm and show that it achieves the order-optimal cumulative regret of  $O(\log(MT))$  while incurring a near-optimal communication cost of  $O(d \log(MT))$  total bits over the entire learning horizon. This algorithm builds upon the insights obtained during the design of PLS and borrows its building blocks from PLS and Minibatch SGD.

The proposed algorithm, referred to as Communication-Efficient Adaptive Learning (CEAL), consists of a decision strategy working in tandem with a communication protocol that ensures the order-optimal regret with low communication cost. The decision strategy is characterized by the norm estimation routine embedded into its design, similar to the PLS algorithm. This novel application norm estimation routine not only allows the algorithm to use high quality estimates of gradient to take larger steps in the direction of the gradient but also *adaptively* tunes the time between communication rounds to ensure low communication frequency. This implicitly ensures that in the initial stage when the iterates are far from the optimum, the algorithm moves away from them quickly with more frequent communication and in the later stages, as the iterates moves closer to the minimum, the algorithm can afford to spend more time on them and communicate less frequently. Moreover, this adaptivity is achieved without any tuning parameters or knowledge of function parameters, making this approach robust to unknown function parameters. This decision strategy is complemented with a communication protocol that ensures low communication frequencies and small message size via quantization and encoding, similar to PLS.

### 5.7.1 Basic Structure of CEAL

The framework underlying CEAL is inspired by that of the popular approach of Minibatch-SGD [89], wherein the same point is queried multiple times between two communication rounds. Specifically, CEAL proceeds in epochs which correspond to the time period between two communications. During an epoch  $k \geq 1$ , all the clients query the same point  $x^{(k)}$  throughout the  $t_k$  time instants of the epoch. At the end of the epoch, the clients compute the sample mean of the observed gradients, quantize it appropriately and send it to the server. The server combines the updates from all the clients and computes the next point as  $x^{(k+1)} = x^{(k)} - \eta \hat{g}_k$  and broadcasts it to the clients after appropriate quantization. Here  $\eta \in (0, 1/5\beta)$  is the step size and  $\hat{g}_k$  denotes the noisy estimate of the gradient obtained by averaging the observations received from the clients. The design objective in CEAL is thus to choose the epoch lengths  $\{t_k\}_{k \in \mathbb{N}}$  along with an associated communication protocol to ensure an order-optimal regret along with a low communication cost.

### 5.7.2 The epoch lengths

The epoch lengths play an important role in ensuring both a low regret and infrequent communication. In order to optimally design the epoch lengths, we first understand their role in achieving the desired performance guarantees. Using the definition of  $x^{(k+1)}$ , the  $\beta$ -smoothness of  $f$ , and the bound on  $\eta$  we can

show that

$$\begin{aligned}
\mathbb{E}[f(x^{(k+1)})] &\leq \mathbb{E}[f(x^{(k)})] + \mathbb{E}[\langle \nabla f(x^{(k)}), x^{(k+1)} - x^{(k)} \rangle] + \frac{\beta}{2} \mathbb{E}[\|x^{(k+1)} - x^{(k)}\|^2] \\
&\leq \mathbb{E}[f(x^{(k)})] - \eta \mathbb{E}[\|f(x^{(k)})\|^2] + \frac{\eta}{2} (\mathbb{E}[\|f(x^{(k)})\|^2] + \mathbb{E}[\varepsilon_k^2]) \\
\mathbb{E}[f(x^{(k+1)})] &\leq \mathbb{E}[f(x^{(k)})] - \frac{\eta}{2} \mathbb{E}[\|\nabla f(x^{(k)})\|^2] + \frac{\eta}{2} \cdot \frac{\sigma^2}{Mt_k}.
\end{aligned} \tag{5.2}$$

If one were to set  $t_k$  to  $2\sigma^2/(M\mathbb{E}[\|\nabla f(x^{(k)})\|^2])$ , then using  $\alpha$ -smoothness of  $f$ , we can conclude that

$$\Delta_{k+1} \leq (1 - \alpha\eta/2)\Delta_k, \tag{5.3}$$

where  $\Delta_k := f(x^{(k)}) - f(x^*)$ . In other words, the sub-optimality gap of the iterates decreases exponentially fast.

This choice of epoch lengths simultaneously offers the benefits of low regret and infrequent communication. Note that this choice of  $t_k$  along with  $\beta$ -smoothness ensure that the regret incurred during each epoch is  $O(1)$  as  $t_k \cdot \sum_{m=1}^M f(x^{(k)}) - f(x^*) = O(t_k \cdot M\|f(x^{(k)})\|^2) = O(1)$ . Moreover, the relation  $\|f(x^{(k)})\|^2 = \Theta(\Delta_k)$  along with Eqn.(5.3) results in exponentially increasing epoch lengths limiting the communication rounds to  $O(\log T)$ .

While such a choice of  $t_k$  achieves the desired performance guarantees, its dependence on the knowledge of  $\|\nabla f(x^{(k)})\|^2$ , which is unknown at the beginning of the epoch, prevents one from setting the epoch length to this predetermined value. To overcome this hurdle, we use the norm estimation routine proposed in Section 5.3.2, which adaptively estimates the norm of an unknown vector  $y$  to within a required accuracy and terminates in  $O(1/\|y\|^2)$  steps with high probability. The integration of this routine with CEAL not only gives the server a more accurate estimate of the gradient, but also allows the epoch lengths to get adaptively and automatically chosen to the required order.

### 5.7.3 The communication strategy

While the sequence  $\{t_k\}_k$  controls the communication frequency between the clients and the server, the communication strategy of CEAL ensures that the messages are small to limit the number of bits transmitted over the channel. Since CEAL builds upon the fundamental idea of PLS of progressively sharing the estimates of vector across epochs, one can use the same communication protocol in CEAL as used for PLS. Thus, CEAL also uses the two stage communication protocol consisting of quantization followed by encoding as used in PLS.

Having specified all the components, we combine all of them together to provide a detailed description of CEAL in Algorithm 14. In the description, the initial point  $x^{(1)}$  is chosen at random from the domain and assumed to be known to all the clients apriori. The epoch index  $k$  corresponds to that of iterates and  $j$  to that of the norm estimation routine. For clarity of notation, we refer to

We set  $s_j$  to  $\lceil 40\sigma^2 \log(16Mj^2/\delta)4^j/M \rceil$ . The length of each epoch,  $t_k$ , is implicitly determined by  $s_j$  as follows. If  $\mathcal{J}_k$  denotes the set of all the different values of  $j$  seen by the algorithm during the  $k^{\text{th}}$  epoch, then  $t_k = \sum_{j \in \mathcal{J}_k} s_j$ . The parameters  $\tau_j$  and  $G_j$  are set to  $3 \cdot 2^{-(j+1)}$  and  $(4\sigma/\sqrt{s_j})(1 + \sqrt{\log(4Mj^2/\delta)/2d})$  respectively. They correspond to bounds on the estimation error at the server and any client respectively.  $B_j$  corresponds to an upper bound on the gradient norm at the end of  $j^{\text{th}}$  epoch and is set to  $\min\{5\tau_{j-1}, 1\}$ . Lastly, the resolution parameter sequences are set to  $\gamma_j := \gamma_0\sigma/\sqrt{s_j}$  and  $\phi_j := \phi_0\tau_j$  for constants  $\gamma_0, \phi_0 \in (0, 1)$ , which are chosen based on communication requirements.

---

**Algorithm 14** Communication-Efficient Adaptive Learning (CEAL)

---

```
1: Input: Initial point  $x^{(1)}$ , step size  $\eta \in (0, 1/\beta)$ 
2: Set  $k, j \leftarrow 1$ 
3: while time horizon is not reached do
4:   For each client  $m$ , take  $s_j$  samples, compute the sample mean  $\hat{g}_j^{(m)}(x^{(k)})$  and
      send  $Q(\hat{g}_j^{(m)}(x^{(k)}), \gamma_j, G_j + B_j)$  to the server
5:   At the server, compute  $\hat{g}_j^{(\text{SERV})}(x^{(k)}) = \frac{1}{M} \sum_{m=1}^M \hat{g}_j^{(m)}(x^{(k)})$ 
6:   if  $\tau_j \leq \|\hat{g}_j^{(\text{SERV})}(x^{(k)})\|_2 / 4$  then
7:     Server broadcasts  $Q(\hat{g}_j^{(\text{SERV})}(x^{(k)}), \phi_j, B_j + \tau_j)$  to all clients
8:     Clients update  $x^{(k+1)} \leftarrow x^{(k)} - \eta Q(\hat{g}_j^{(\text{SERV})}(x^{(k)}), \phi_j, B_j + \tau_j)$ 
9:      $k \leftarrow k + 1$ ,
10:    else
11:       $j \leftarrow j + 1$ 
12:    end if
13: end while
```

---

### 5.7.4 Performance Analysis

We characterize the performance of CEAL in terms of cumulative regret and the communication cost it incurs, beginning with bounding the regret incurred by CEAL.

#### Regret analysis

The following theorem characterizes the cumulative regret incurred by CEAL.

**Theorem 5.7.1.** *Consider a distributed stochastic optimization setup as described in Section 5.7. If CEAL is run with parameters described above, then the cumulative regret incurred by CEAL is bounded by  $O(\log(MT) \log(M/\delta))$  with probability at least  $1 - \delta$ .*

Theorem 5.7.1 establishes the regret performance of CEAL. Note that it matches with the lower bound for any algorithm in a centralized setting with

$MT$  queries implying that the regret performance of CEAL is indeed order-optimal.

*Proof.* To bound the regret incurred by CEAL, note that in each epoch  $k$ , each client queries the point  $x^{(k)}$  for a total of  $t_k$  times. Consequently, the regret incurred by CEAL can be written as

$$\begin{aligned} R(T) &= \sum_{m=1}^M \sum_{k=1}^K (f(x^{(k)}) - f(x^*)) \cdot t_k \\ &\leq 2\beta M \sum_{k=1}^K \|\nabla f(x^{(k)})\|^2 \cdot t_k, \end{aligned} \quad (5.4)$$

where  $K$  is the (random) number of epochs carried out during the execution of the algorithm. The following lemma provides a bound on the length of the  $k^{\text{th}}$  iteration,  $t_k$ .

**Lemma 5.7.2.** *Suppose CEAL is run with parameters described above. If it queries a point  $x^{(k)}$  during epoch  $k$ , then the length of epoch  $k$ ,  $t_k$ , as defined in Section 5.7.2 satisfies  $\Theta(1/(M\|\nabla f(x^{(k)})\|^2))$  with probability at least  $1 - \delta$ .*

In addition to the above bound, the iteration lengths also satisfy the constraint  $\sum_{k=1}^K t_k \leq T$  as each client cannot issue more than  $T$  gradient queries. Consequently, the upper bound on regret is the value of the following constrained maximization problem which is obtained by using bounds on  $t_k$  described in Lemma 5.7.2.

$$\begin{aligned} \max \quad & 2\beta \sum_{k=1}^K \left[ 4320\sigma^2 \cdot \log \left( \frac{16M}{\delta} \log^2 \left( \frac{9}{2\|\nabla f(x^{(k)})\|} \right) \right) + M\|\nabla f(x^{(k)})\|^2 \left( \log \left( \frac{9}{2\|\nabla f(x^{(k)})\|} + 1 \right) \right) \right] \\ \text{s.t. } & \sum_{k=1}^K \left[ \frac{4320\sigma^2}{M\|\nabla f(x^{(k)})\|^2} \cdot \log \left( \frac{16M}{\delta} \log^2 \left( \frac{9}{2\|\nabla f(x^{(k)})\|} \right) \right) + \left( \log \left( \frac{9}{2\|\nabla f(x^{(k)})\|} + 1 \right) \right) \right] \leq T. \end{aligned}$$

On applying the method of Lagrange Multipliers, one can immediately conclude that value of the aforementioned constrained problem is  $O(K \log(M/\delta))$ ,

i.e., proportional to the number of iterations carried out during the algorithm. The following lemma provides high probability bound on the number of epochs during a execution of CEAL.

**Lemma 5.7.3.** *If CEAL is run with parameters described above, then both the total number of epochs during its run is bounded by  $O(\log(MT))$  with probability at least  $1 - \delta$ .*

The theorem now follows immediately by noting that  $R(T)$  is  $O(K)$  and invoking the above lemma. Since both Lemma 5.7.2 and 5.7.3 provide high probability bounds, the resultant bound on the regret incurred by CEAL also holds with probability at least  $1 - \delta$ .

□

Please refer to Appendix D for proofs of Lemmas 5.7.2 and 5.7.3.

## 5.8 Empirical Studies

In this section, we provide empirical evidence that corroborates our theoretical findings. In particular, we compare our proposed algorithms, PLS and CEAL, with popular distributed bandit and Federated Learning algorithms respectively and establish their superior performance, in terms of both regret and communication cost.

### 5.8.1 PLS

We compare our proposed PLS algorithm with three popular distributed linear bandit algorithms, namely, Distributed Elimination for Linear Bandits (DELB) [320], Federated Phased Elimination (Fed-PE) [143] and Distributed Batch Elimination Linear Upper Confidence Bound (DisBE-LUCB) [11].

We consider a distributed linear bandit instance with  $d = 20$ ,  $M = 10$  agents which is run for a time horizon of  $T = 10^6$  steps. The underlying mean reward vector is drawn uniformly from the surface of a unit ball. The rewards are corrupted with a zero mean Gaussian with unit variance. We plot the averaged cumulative regret for different algorithms considered over the time horizon of  $T$  in Fig. 5.1 and report the uplink and downlink communication costs in Table 5.1. Recall that the uplink communication cost is defined as the number of bits sent by *one* agent to the server and the downlink cost is defined as the number of bits broadcast by the server. All the results are reported after averaging over 10 Monte Carlo runs. For a fair comparison, we assume that the real numbers are represented by  $\log(MT)$  bits, which comes out to 24 bits in our setting as opposed to 32 for regular floats. We, however, for the purpose of implementation transfer the full float representation, which works in favor of the other algorithms.

### Experimental Setup

Since DELB, Fed-PE and DisBE-LUCB are designed for different settings, we perform a pairwise comparison of PLS with each of these algorithms based on the setting for which they are original designed for. We describe each of experi-

mental setups in detail below.

- DELB: Since the underlying setting in DELB is the same as that considered in PLS, we carry out two sets of experiment to compare the performance of PLS against that of DELB. In the first experiment, we consider a linear bandit instance with unit ball as the action space. In the second experiment, we consider an action space consisting of  $K = 120$  actions drawn from a unit ball at random.
- FED-PE: We consider the shared parameter setting described in Huang *et al.* [143]. For each agent, we choose  $K = 120$  actions randomly from the unit ball, independent of other agents. The phase lengths for FED-PE are set to be growing in powers of 2, similar to the choice adopted in Huang *et al.* [143].
- DisBE-LUCB: We adopt the same experimental setup as considered in Amani *et al.* [11] for the evaluation of DisBE-LUCB. However, instead of a distribution over 100 instances, we consider a distribution over 50 instances with  $K = 40$  actions in each of them.

Depending on the experimental setup, we either use the implementation of PLS as outlined in the main paper or that of its extension to general action spaces as described in Section 5.6.2.

## Results

PLS offers a significantly lower cumulative regret as compared to DELB in both the cases and the significant improvement of PLS over DELB in terms of communication cost also evident from Table 5.1. In particular, the difference in

Experiment	Algorithm	Uplink Cost	Downlink Cost
DELB continuous action space	DELB	531.8	7400.2
	PLS	103.0	139.6
DELB discrete action space	DELB	336.0	4700
	PLS	112.9	155.1
Federated homogeneous setting	Fed-PE	72960	249900
	PLS	301.1	402.5
Stochastic Contexts	DisBE-LUCB	2000	2000
	PLS	188.9	309.5

Table 5.1: Communication cost (in bits) for various algorithms against PLS in their corresponding experimental setup. Reported values are obtained after averaging over 10 Monte Carlo runs.

downlink cost is significant due to the linear scaling with the number of agents for DELB. For the case for FED-PE, PLS outperforms it in terms of regret incurred despite not being designed for this heterogeneous setting. In terms of communication cost, the scaling with respect to the number of actions significantly deteriorates the uplink cost for FED-PE and the dependence on  $d^2$  worsens the downlink cost. On the other hand, PLS continues to enjoy both small uplink and downlink costs. Lastly, PLS also offers superior performance over DisBE-LUCB, both in terms of regret and communication cost, even within the stochastic contextual bandit setup. The above experimental results demonstrate the improved performance of PLS over existing distributed linear bandit algorithms across a variety of setups.

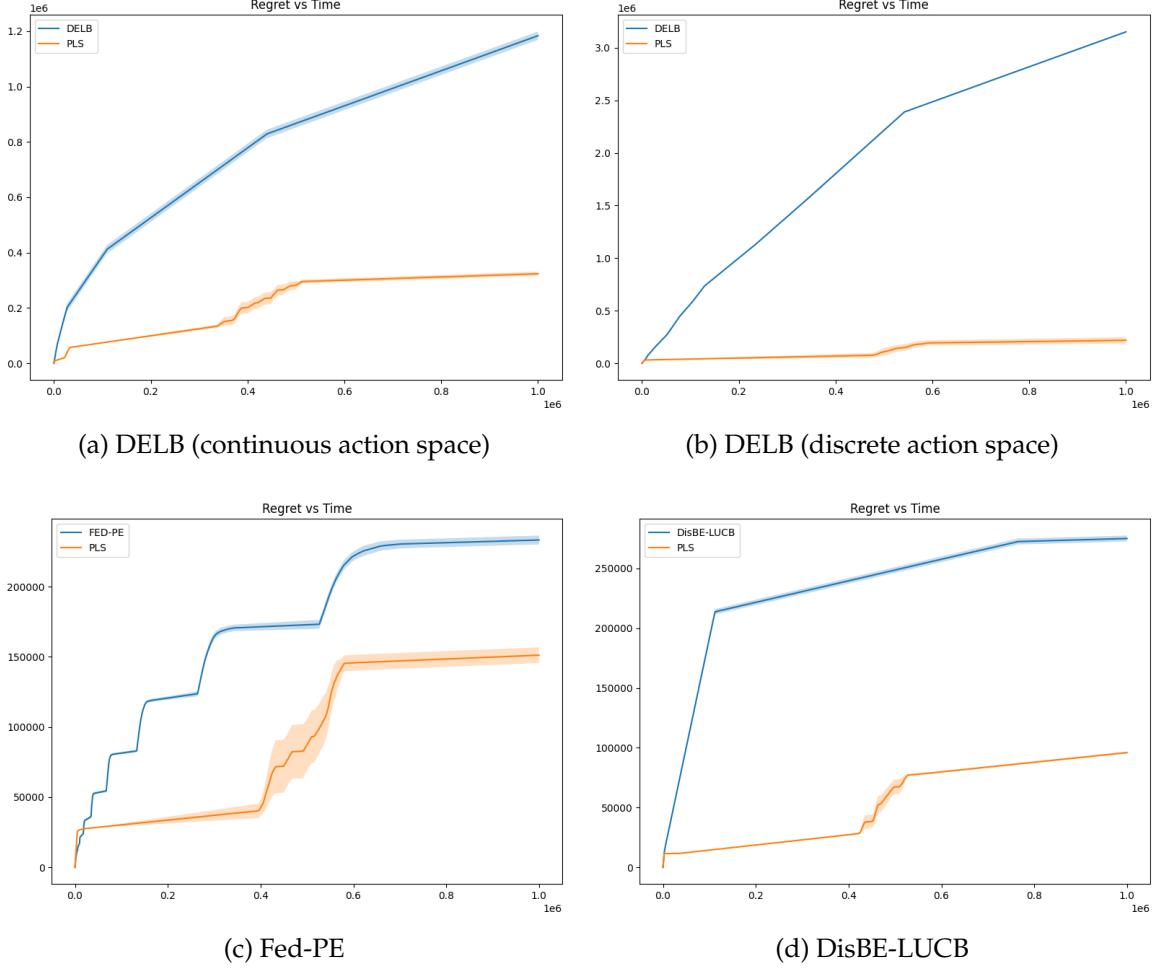


Figure 5.1: Cumulative Regret vs Time for different algorithms. The bold line represents the mean obtained over 10 Monte Carlo runs and the shaded region represents the region of error bars corresponding to one standard deviation.

### 5.8.2 CEAL

In this section, we provide numerical experiments comparing the performance of CEAL with that

We compare the performance of CEAL against several baselines, namely, Minibatch-SGD [329], FedAvg [207], FedPAQ [232] and FedCOM [122]. We first describe the datasets used in the experiments and experimental setup, followed by a discussion on the results.

## Datasets

We perform empirical studies on both synthetic and real-world datasets. For the synthetic dataset, we consider the problem of linear regression where the loss (objective) function is given by  $f(\theta) = \frac{1}{N} \sum_{i=1}^N (y_i - X_i^\top \theta)^2$ , where  $\theta \in \mathbb{R}^{30}$ . The covariates  $\{X_i\}_{i=1}^N$  are drawn from zero mean normal distribution and are normalized and scaled to have norm of 100. The responses  $\{y_i\}_{i=1}^N$  are generated as  $y_i = X_i^\top \theta^* + \varepsilon_i$ , where  $\varepsilon_i$ 's are independent and identically distributed as  $N(0, 1)$  and  $\theta^*$  is the true unknown set of regression coefficients and is drawn uniformly at random from the surface of the unit sphere. The number of data points are set to  $N = 2000$  and are distributed uniformly across the  $M = 10$  devices, with each getting 200 data points.

We also consider the problem of regularized logistic regression on MNIST dataset. From the original training dataset of 60,000 images, we consider a subset of 50,000 images, with 5,000 images corresponding to each digit. This dataset is uniformly distributed across all  $M = 10$  clients, resulting in 5,000 data points for each client. The images are normalized to ensure that the pixel values lie in  $[0, 1]$ . We consider the standard loss function for multinomial logistic regression given by

$$f(W) = \frac{1}{N} \sum_{i=1}^N \left[ \sum_{k=0}^9 \mathbb{1}\{Y_i = k\} X_i^\top W + \log \left( \sum_{k=0}^9 \exp(-X_i^\top W_k) \right) \right] + \mu \|W\|_F^2$$

Here  $W \in \mathbb{R}^{784 \times 10}$  is a weight matrix for classification,  $X_i \in \mathbb{R}^{784}$  is the vectorized training data,  $Y_i$  is the corresponding label,  $k \in \{0, 1, \dots, 9\}$  denotes the class,  $N = 50,000$  is the number of data points,  $\mu > 0$  is the regularization parameter and  $\|\cdot\|_F$  denotes the Frobenious norm.

## Experimental Settings

For both synthetic and real datasets, we consider a Federated Learning problem with  $M = 10$  devices connected to a central server. All the algorithms are run for a time horizon of  $T = 2000$  steps for the synthetic dataset and  $T = 1000$  for the MNIST dataset. The gradient is computed using a randomly chosen minibatch for each client, based on the each client’s data. For the synthetic and MNIST datasets, the minibatch size is set to 1 and 25 respectively. The learning rate for all the algorithms is optimized using a grid search. For the synthetic dataset, the learning rate of Minibatch-SGD, FedAvg, FedPAQ, FedCOM<sup>5</sup> and CEAL were set to 1, 0.1, 0.1, 0.002 and 2 respectively. Similarly, for the MNIST dataset the learning rates (in the same order) were set to 0.2, 0.01, 0.01, 0.0005 and 0.3 respectively. For the synthetic dataset, the number of local steps was set to 100 for FedAvg, FedPAQ and FedCOM and 50 for Minibatch SGD. For the real dataset, the number of local steps was set to 50 for all algorithms. The regularization parameter was set to 0.5.

We assume 32 bit representation of floats in order to calculate the communication cost for algorithms that do not employ quantization. The number of quantization levels for FEDPAQ and FEDCOM is set to 3 for synthetic dataset and 5 for real dataset. We report the cumulative regret and the communication cost (both uplink and downlink), measured in bits for all the algorithms, averaged over 10 Monte Carlo runs. Recall that the uplink and downlink cost were defined to the number of bits transmitted by a client (on average) to the server and the those broadcast by the server to the clients throughout the entire learning process.

---

<sup>5</sup>The global learning rate was set to 10 in both the experiments.

## Results

We plot the overall cumulative regret incurred by different algorithms for the experiment with synthetic dataset in Fig. 5.2a and that for the real dataset in Fig. 5.2b. We tabulate the communication costs incurred by different algorithms across different experiments in Table 5.2. As it is evident from the plots, our proposed algorithm achieves a smaller cumulative regret than the standard, commonly used algorithms for Federated Learning for both the tasks. Moreover, this improved performance in regret is achieved at a very low communication cost. Specifically, for synthetic and real dataset the uplink communication cost incurred by CEAL is about 10% and 16% respectively of that incurred by FedPAQ and FedCOM algorithms, both of which popular Federated Learning algorithms that employ quantization to reduce communication. This factor reduces to 1 – 2% when compared against classical algorithms like FedAvg that do not employ quantization. CEAL also offers significant reduction in downlink costs as it incurs no more than 2% of the downlink cost incurred by all other algorithms. This can significantly reduce the download costs for local training devices and consequently reduce the infrastructure requirements for local participating devices.

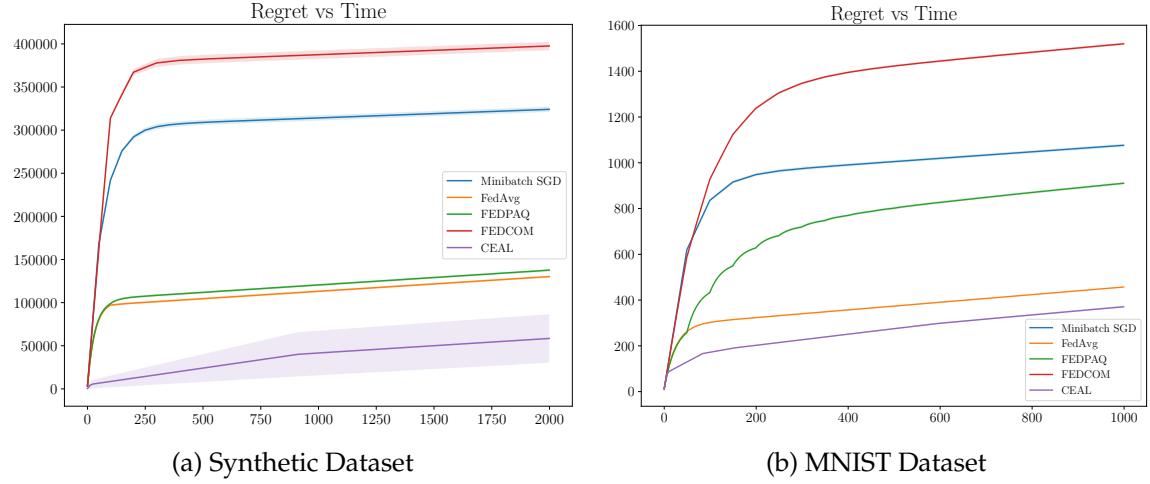


Figure 5.2: Cumulative Regret vs Time for different algorithms for on (a) Synthetic dataset and (b) MNIST dataset. The bold line represents the mean obtained over 10 Monte Carlo runs and the shaded region represents the region of error bars corresponding to one standard deviation. The error bars for all algorithms in on MNIST are very small (approximately  $\pm 10$ ).

	Uplink Cost		Downlink Cost	
	Synthetic Dataset	MNIST	Synthetic Dataset	MNIST
Minibatch SGD	38400	$5.02 \times 10^6$	38400	$5.02 \times 10^6$
FedAvg	19200	$5.02 \times 10^6$	19200	$5.02 \times 10^6$
FedPAQ	2440	$0.63 \times 10^6$	19200	$5.02 \times 10^6$
FedCOM	2440	$0.63 \times 10^6$	19200	$5.02 \times 10^6$
CEAL	263.3	$0.11 \times 10^6$	288.6	$0.26 \times 10^6$

Table 5.2: Communication cost (in bits) for various algorithms on different datasets. Reported values are obtained after averaging over 10 Monte Carlo runs.

CHAPTER 6

**DISTRIBUTED LEARNING AMONG PROBABILISTICALLY  
HETEROGENEOUS USERS**

## 6.1 Introduction

### 6.1.1 Kernel-based Bandit Problems

We consider zeroth-order stochastic optimization where the objective function is unknown but assumed to live in a Reproducing Kernel Hilbert Space (RKHS) associated with a known kernel. This zeroth-order stochastic optimization problem can be equivalently viewed as a continuum-armed kernelized-bandit problem [154]. The RKHS model allows for a greater expressive power for representing a broad family of objective functions. In particular, it is known that the RKHS of typical kernels, such as Matérn family of kernels, can approximate almost all continuous functions on compact subsets of  $\mathbb{R}^d$  [278]. Under a centralized setting where all data is available at a single decision maker, several algorithms have been proposed as discussed in Chapter 3, including UCB-based algorithms [278, 68], batched pure exploration [193], and tree-based domain shrinking [247].

In recent years, distributed stochastic optimization with multiple clients has witnessed growing interest and is encountered in numerous applications. For example, in a federated learning setting of the text prediction problem, multiple mobile users aim to learn a text predictor by leveraging each other's local data but without sharing their raw data [189]. Another example, more specific to the

kernelized bandit setting, is hyperparameter tuning in federated learning [78]. Due to the large expressive power of RKHS functions, hyperparameter tuning in centralized settings is often carried out by modelling the accuracy of the underlying neural network as an unknown function of the hyperparameters and belonging to an RKHS. However, when a model is being learned by a collective effort of multiple clients, the hyperparameters of the model often also need to be tuned collaboratively using the information from all clients, resulting in a distributed stochastic optimization problem of a RKHS function. An interesting application of distributed optimization of RKHS functions is in the analysis of the collaborative training of neural nets using the recent theory of Neural Tangent Kernel [147].

Comparing with the centralized setting, distributed stochastic optimization gives rise to two new challenges. First, the probabilistic model underlying local data generation may be different across clients, resulting in different local objective functions and hence the pursuit of different optimizers at distributed clients. Consider for example the text prediction problem. The probabilistic dependency of the next word on the past typed words may exhibit discrepancies caused by geographical or cultural differences across users. This drift in data distribution calls for learning algorithms that strike a balance between data sharing and targeted local performance. In particular, how heterogeneous clients can learn collectively by leveraging each other’s data while at the same time converging to an optimizer that takes into account each client’s local data model. The prevailing approach where all clients aim to optimize a common global objective function given by a weighted sum of each client’s local objectives with prefixed weight operates at one extreme of the spectrum by focusing solely on reaping the benefit of data sharing while ignoring each client’s local

performance. The second challenge, discussed in the previous chapters, is the communication overhead associated with information exchange across clients. While communication is necessary for collaborative learning, low communication cost is of practical importance in many applications. These challenges have not been adequately addressed in the literature, especially within the kernel-based distributed bandit framework (see Section 6.1.3 for a detailed discussion of related work).

### 6.1.2 Main Contributions

Consider a distributed setting with  $K$  clients connected to a central server that facilitates information exchange, a de facto setting for training large-scale machine learning models. Each client can sequentially query its local objective function (i.e., the expected reward with respect to its local data model) and observe an unbiased noisy estimate of the function value at the chosen query point. To handle heterogeneity in local data models, we adopt the personalization framework, wherein the objective of each client is to learn a mixture of their local function and the global function, where the global objective function is an average of all local functions. The personalization framework strikes a balance between the generalization capabilities of the global function and targeted local learning tailored towards each client’s local model. By allowing the full range of the mixture weights (i.e., the personalization parameters), we address the full spectrum in terms of the balance between data sharing and targeted local performance.

Within this personalization framework, each client is interested in maxi-

mizing its own *personalized objective function*, which differs from those of other clients. The challenge here is that the personalized objectives  $f_k$  at each user  $k$  involves other users' local functions  $\{h_j\}_{j \neq k}$  which cannot be observed by user  $k$ . In other words, the local observations at each end user  $k$  provide only a partial view of the personalized objective  $f_k$  it aims to optimize. This necessitates collaborative learning across users in order to achieve a sublinear regret order. The required user collaboration has two aspects: information sharing and collaborative exploration. Given that  $h_j$  can only be queried at user  $j$ , it is easy to see that information sharing is necessary for each user  $k$  to learn its personalized objective  $f_k$ . The need for collaborative exploration is a more nuanced and complex issue. Consider the extreme case where every user's local observations are made available to all other users. If user  $k$  simply uses a centralized learning algorithm to optimize its own personalized objective function  $f_k$ , its local model may concentrate too quickly around its own optimal point, resulting in insufficient exploration of  $h_k$  around optimal points of the personalized local models of other users, especially for a user whose optimal point is far away from that of user  $k$ . More specifically, even though each user aims to converge to different optimal models, collaborative exploration—in addition to information sharing—is necessary due to the coupling across users' personalized objective functions. This adds another layer of nuance to the exploration-exploitation trade-off as the exploration at every user needs to serve the goal of greater good at the network level instead of solely being sufficient for local exploitation.

We propose a new learning algorithm for kernelized distributed bandits with probabilistically heterogeneous clients. Referred to as Collaborative Exploration with Personalized Exploitation (CEPE), this algorithm is built on the key structure of interleaving exploration epochs for the collaborative learning

of the global function with exploitation epochs aiming to maximize individual local performance. This interleaving exploration-exploitation structure not only addresses the trade-off between of data leveraging and targeted local performance, but also effectively controls the information exchange across clients for communication efficiency through a tandem design of the accompanying communication protocol. The lengths of the exploration and exploitation epochs are carefully controlled to balance learning efficiency and communication overhead associated with information exchange.

We analyze the performance of the proposed algorithm and show that its regret is of  $\tilde{O}(KT^{\frac{2}{3-\kappa}})$ <sup>1</sup> and its overall communication cost is of  $O(T^{\frac{2}{3-\kappa}})$ , where  $\kappa \in (0, 1]$  is a parameter that depends on the smoothness of the kernel through its information gain. We further establish a lower bound on achievable regret within the kernel-based collaborative learning framework for the class of Squared Exponential and Matern kernels, which are arguably the most widely used kernels in practice. The regret lower bound matches the order of the regret incurred by CEPE up to a poly-logarithmic factor, establishing the optimality of the proposed algorithm. Empirical studies demonstrate the effectiveness of the CEPE against several baseline algorithms, bolstering our theoretical results.

To further reduce the communication cost, we propose a variant of CEPE which employs sparse approximations of the surrogate Gaussian Process modelling used in the query and update rules of CEPE. Referred to as S-CEPE, this algorithm reduces the communication cost to  $O(T^{\frac{2\kappa}{3-\kappa}})$  while preserving the regret guarantee of CEPE. Numerical examples demonstrate effectiveness of S-CEPE in practice as S-CEPE is shown to offer a 14-fold reduction in communication cost while sacrificing regret by a factor of less than 2 when compared to CEPE.

---

<sup>1</sup> $\tilde{O}(\cdot)$  hides the polylogarithmic factors.

### 6.1.3 Related Work

Kernel-based bandit problem in the centralized setting has been extensively studied in the literature [68, 278, 308, 135, 291, 50, 260]. The setup considered in these studies, where all the data is available at a central server, is inherently different from the distributed setup considered in this work. Please refer to Remark 6.3.2 for additional comparison with these results.

Various studies have considered the discrete multi-armed bandit problem and linear bandit problem in the distributed setting. Shi and Shen[269] and Shi *et al.* [270] consider the multi-armed bandit (MAB) problem in a distributed setup with homogeneous and heterogeneous client reward distributions respectively. The authors in [190] considered the same problem with distributionally identical clients, with a focus on ensuring privacy. Hillel et al. [136] consider the pure exploration problem for multi-armed bandits connected over a network. Other representative works in the MAB setting include [198, 262, 176, 254]. Wang *et al.* [320] proposed the DELB algorithm for the distributed linear bandit setting with focus on communication efficiency. Dubey *et al.* [101] proposed a privacy preserving algorithm for distributed linear bandits with clients connected via various network topologies. Other studies exploring linear bandits in a distributed setup include [171, 143, 115, 11, 251]. The kernel-based bandit setting is arguably more challenging than the classical MAB or the linear bandit setting.

The problem of kernel-based bandits in collaborative setups has not received sufficient attention. There exist only a handful studies. Du *et al.* [99] considered the problem of pure exploration in kernel-based collaborative learning over a finite action set. Another work in this direction is by Li *et al.* [187] where the au-

thors propose a communication-efficient algorithm with a communication cost of  $O(\gamma_T^3)$ . Dubey *et al.* [102] consider distributed kernel bandits over a graph where they consider additional kernel based modelling to measure task similarity among the different objective functions of the clients. However, their proposed algorithm suffers from a communication cost that grows linearly with the time horizon. In this work, we consider the more challenging continuum-armed setup with a focus on minimizing cumulative regret as opposed to simple regret. Moreover, the personalization framework considered in this work is different from the other studies in kernel-based collaborative learning. Furthermore, the S-CEPE algorithm proposed in this work incurs a communication cost of  $O(T^{\frac{2\kappa}{3-\kappa}})$ , improving upon the  $O(T^{3\kappa})$  bound obtained for the algorithm in [187].

The problem of distributed first-order stochastic convex optimization, often referred to as Federated Learning (FL) [207], is yet another direction of related work that has been gaining a lot of attention in the recent times. See [4, 152] for a survey of recent advances. Several works in FL have also considered the personalization framework [272, 150, 316, 91, 106, 131, 173, 204] as considered in this work. Despite some similarities in the setup, the zeroth-order kernel-based setup considered in this work requires very different tools and techniques for algorithm design and analysis as compared to the first-order stochastic convex optimization problem considered in FL.

## 6.2 Problem Formulation and Preliminaries

In this section, we present the problem formulation followed by preliminaries on Gaussian Processes (GPs), an important tool in the design of the proposed

algorithm.

### 6.2.1 Problem Formulation

We consider a collaborative learning framework with a star topology consisting of a central server and  $K > 1$  clients. Each client  $i \in \{1, 2, \dots, K\}$  is associated with a local *observation* function,  $h_i : \mathcal{X} \rightarrow \mathbb{R}$ , which it can access by querying any point  $x$  in the domain  $\mathcal{X} \subset \mathbb{R}^d$  and consequently receiving a noisy evaluation  $y = h_i(x) + \epsilon$ , where  $\epsilon$  is the noise term. These observation functions  $h_k$  are known to live in the Reproducing Kernel Hilbert Space (RKHS) associated with a known positive definite kernel  $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ . We make the following assumptions on the observation functions and noise, commonly adopted in kernel-based learning [278, 68].

**Assumption 6.2.1.** The RKHS norm of  $h_i$  is bounded by a known constant  $B_i$ :  
 $\|h_i\|_{H_k} \leq B_i$ , for each  $i \in \{1, 2, \dots, K\}$ .

**Assumption 6.2.2.** The noise terms  $\epsilon$  are assumed to be zero mean  $R$ -sub-Gaussian variables, i.e.,  $\mathbb{E}[e^{\zeta\epsilon}] \leq \exp(\zeta^2 R^2/2)$  for all  $\zeta \in \mathbb{R}$ , that are distributed independently across queries and clients.

**Assumption 6.2.3.** For each given  $n \in \mathbb{N}$  and  $f \in H_k$  with  $\|f\|_{H_k} \leq B$ , there exists a discretization  $\mathcal{D}_n$  of  $\mathcal{X}$  such that  $|f(x) - f([x]_{\mathcal{D}_n})| \leq 1/\sqrt{n}$ , where  $[x]_{\mathcal{D}_n} = \arg \min_{y \in \mathcal{D}_n} \|x - y\|_2$  is the closest point in  $\mathcal{D}_n$  to  $x$  in terms of  $\ell_2$  distance and  $|\mathcal{D}_n| \leq CB^d n^{d/2}$ , where  $C$  is a constant independent of  $n$  and  $B$ .

## Personalized Reward Function

In a typical collaborative learning framework, the clients work together to optimize a common objective function, referred to as the *global* reward function, which is defined as,

$$g(\cdot) := \frac{1}{K} \sum_{i=1}^K h_i(\cdot). \quad (6.1)$$

For example, Shi and Shen [269] considered the finite-armed bandit problem with the above mentioned objective. In a more general case, the weights  $1/K$  can be replaced by any weight distribution over the  $K$  clients. In typical collaborative learning problems, the motivation behind optimizing such a *global* function is to leverage information across clients. The availability of additional data facilitated by the participation of more clients helps in learning a better model that generalizes well. In the context of this work, the argument of  $g$  corresponds to the parameters of the model and consequently, the maximizer of  $g$  corresponds to the desired optimal parameters.

While collaboratively optimizing  $g$  can help leverage information across clients, optimizing the global objective  $g$  may not always faithfully represent the interests or desires of the clients, especially in the scenario of statistically heterogeneous clients. In contrast, it can be detrimental for a client to only stick to model obtained using local data and forgo the additional generalization capabilities offered by collaboration. Following [91, 131, 270], we consider a collaborative learning model with personalization, where each client can choose to trade-off between the generalization capabilities of the global reward function and a locally focused reward function that aims to fit local data only. In

particular, we consider a personalized *reward* function for each client  $i$  given by,

$$f_i(\cdot) = \alpha_i h_i(\cdot) + (1 - \alpha_i)g(\cdot), \quad (6.2)$$

which is a linear combination of the global reward and the local observation functions. Here, the personalization parameter  $\alpha_i \in (0, 1)$  captures the level of personalization for each client  $i$ . A choice of  $\alpha_i \rightarrow 0$  corresponds to adopting a fully collaborative approach. Increasing the value of  $\alpha_i$  reduces the impact of collaboration and as  $\alpha_i \rightarrow 1$ , the client moves towards adopting a model that solely targets local observations, forgoing all benefits of collaboration. In this work, we focus on the cases  $\alpha_i \in (0, 1)^2$  which corresponds to the scenario where clients have different objective functions and need to collaborate while having different personalized objectives. We would like to point out that this personalized setting is more challenging than the scenario where all clients collaboratively optimize a *common* global objective (all  $\alpha_i = 0$ ) or all of them are interested in optimizing their local observation functions (all  $\alpha_i = 1$ ). For clarity of terminology, we refer to  $g$  as the global reward,  $h_i$  as the local observation, and  $f_i$  as the personalized reward for client  $i$ .

## Communication Protocol and Cost

Each personalized reward  $f_i$  depends on the observation functions  $h_j$  of other clients  $j \neq i$  which are cannot be accessed by client  $i$ . The clients thus need to share information about their local observations with each other, in order to make learning  $f_i$  feasible and incur a sublinear regret. In our setup, the clients can communicate only with central server and cannot communicate with other clients, a typical setting in networks with a star topology. At each time  $t$ , each

---

<sup>2</sup>This condition can be relaxed to  $\max_i \alpha_i > 0$  and  $\min_i \alpha_i < 1$ .

client  $i$  is allowed to send and receive a message to and from the server. For simplicity, we assume that all the communication is synchronized, a commonly adopted assumption in collaborative learning frameworks. We assign a unit cost to each communication of a scalar from the client to other clients through the server. This definition of communication cost is different from holistic definition adopted in the previous chapters. We adopt this definition to focus more on the statistical heterogeneity aspect of the problem. Designing algorithms that address statistically heterogeneous clients while also achieving minimal order communication cost at the bit level is an interesting future direction.

### The learning objective

A collaborative learning policy  $\pi = \{\pi_{i,t}\}_{t \geq 1, i=1,2,\dots,K}$  specifies, for each client  $i$ , which point  $x_{i,t}$  to query at each time  $t$  based on available information. The performance of  $\pi$  is measured in terms of total cumulative regret summed over all clients and over a learning horizon of length  $T$ . Specifically,

$$R_\pi(T) = \sum_{i=1}^K \sum_{t=1}^T (f_i(x_i^*) - f_i(x_{i,t})), \quad (6.3)$$

where  $x_i^* = \arg \max_{x \in \mathcal{X}} f_i(x)$  is the optimal decision variable for client  $i$ 's personalized reward function  $f_i$ .

The objective is to design a policy  $\pi$  which minimizes the total cumulative regret. We provide high probability regret bounds that hold with probability at least  $1 - \delta_0$  for any given  $\delta_0 \in (0, 1)$ , a stronger performance guarantee than bounds on expected regret.

### 6.2.2 Preliminaries on GP Models

In this section, we overview the GP models and some useful confidence intervals for the RKHS elements based on GP models, which are central to our policy design. This section is largely a recap of the discussion in Section 3.1.1 and has been added here for completeness.

A Gaussian Process model,  $H(x)$ ,  $x \in \mathcal{X}$  is a random process indexed on  $\mathcal{X}$ , for which all finite subsets  $\{H(x_i)\}_{i=1}^n$ ,  $n \in \mathbb{N}$ , have a multivariate Gaussian distribution, with mean  $\mu(x) = \mathbb{E}[H(x)]$  and covariance  $k(x, x') = \mathbb{E}[(H(x) - \mu(x))(H(x') - \mu(x'))]$ . We assume  $\mu(x) = 0$ , for all  $x \in \mathcal{X}$ . When used as a prior for a data generating process under Gaussian noise, the conjugate property provides closed form expressions for the posterior mean and covariance of the GP model. In particular, given a set of observations  $\mathcal{H}_t = \{\mathbf{x}_t, \mathbf{y}_t\}$ , where  $\mathbf{x}_t = (x_1, x_2, \dots, x_t)^\top$ ,  $\mathbf{y}_t = (y_1, y_2, \dots, y_t)^\top$ , the following expressions can be derived for the posterior mean and covariance of the GP model

$$\mu_t(x) = \mathbb{E}[H(x)|\mathcal{H}_t] = k_{\mathbf{x}_t, x}^\top (K_{\mathbf{x}_t, \mathbf{x}_t} + \lambda I_t)^{-1} \mathbf{y}_t \quad (6.4)$$

$$\begin{aligned} k_t(x, x') &= \mathbb{E}[(H(x) - \mu_t(x))(H(x') - \mu_t(x'))|\mathcal{H}_t] \\ &= k(x, x') - k_{\mathbf{x}_t, x}^\top (K_{\mathbf{x}_t, \mathbf{x}_t} + \lambda I_t)^{-1} k_{\mathbf{x}_t, x'}. \end{aligned} \quad (6.5)$$

In the above expressions,  $k_{\mathbf{x}_t, x} = [k(x_1, x), \dots, k(x_t, x)]^\top$ ,  $K_{\mathbf{x}_t, \mathbf{x}_t}$  is the  $t \times t$  covariance matrix  $[k(x_i, x_j)]_{i,j=1}^t$ ,  $I_t$  is the  $t \times t$  identity matrix and  $\lambda$  is the variance of the Gaussian noise. We use  $\sigma_t^2(\cdot) = k_t(\cdot, \cdot)$  to denote the posterior variance of the GP model.

Following a standard approach in the literature (e.g., see [68, 267, 278]), we use GPs to model an unknown function  $h$  belonging to the RKHS corresponding to the covariance kernel of the GP. In particular, we assume a *fictitious* GP prior

$H$  over the *fixed*, unknown function  $h$  along with *fictitious* Gaussian distribution for the noise. We would like to emphasize that these assumptions are modelling techniques used as a part of algorithm and not a part of the problem setup. The reason for introducing this fictitious model is that the posterior mean and variance defined above, respectively, provide powerful tools to predict the values of  $h$ , and to quantify the uncertainty in the prediction. We formalize this statement in the following lemma.

**Lemma 6.2.4** (Theorem 1 in [299]). *Under Assumptions 6.2.1 and 6.2.2, provided observations  $\mathcal{H}_t = \{\mathbf{x}_t, \mathbf{y}_t\}$  as specified above with the query points  $\mathbf{x}_t$  chosen independently of the noise sequence, we have, for a fixed  $x \in \mathcal{X}$ , with probability at least  $1 - \delta$ ,*

$$|h(x) - \mu_t(x)| \leq \beta(B, \delta) \sigma_t(x),$$

where  $\beta(B, \delta) = B + R \sqrt{(2/\lambda) \log(2/\delta)}$ .

This lemma shows that  $\mu_t$  may be used to predict the value of  $h$ , where the error in the prediction is bounded by a factor of  $\sigma_t$  with high probability. In CEPE,  $\sigma_t(\cdot)$  is used to guide exploration and  $\mu_t(\cdot)$  is used to guide exploitation.

Lastly, we define a kernel dependent term called maximal information gain,  $\gamma_t$ , which characterizes the effective dimension of the kernel. It is defined as  $\gamma_t := \max_{\mathbf{x}_t=(x_1, x_2, \dots, x_t) \in \mathcal{X}^t} \frac{1}{2} \log(I + \lambda^{-1} K_{\mathbf{x}_t, \mathbf{x}_t})$ , where  $I$  is a  $t \times t$  identity matrix. Bounds on  $\gamma_t$  for several common kernels are known [279, 301] and are increasing sublinear functions of  $t$ , i.e.,  $\gamma_t = O(t^\kappa)$  for  $\kappa \in (0, 1]$ . Thus,  $\kappa$  characterizes the order of the growth of the effective dimension of the kernel with the number of points.

## 6.3 The CEPE Policy

In this section, we describe our proposed algorithm for the problem of personalized kernel bandits.

### 6.3.1 Algorithm Description

The design of a collaborative learning policy such as CEPE significantly deviates from those for single client settings. If one were to design a collaborative learning algorithm by simply deploying a classical centralized kernel-based learning algorithm like GP-UCB or GP-TS [68] at each client to learn their personalized rewards, then such an algorithm could result in a trivial regret, linearly growing with  $T$ . This can be attributed to the fact that the personalized objective function  $f_i$  of client  $i$  contains the local observation functions  $\{h_j\}_{j \neq i}$  of other clients that are not observable to client  $i$ . If client  $i$  were to use GP-UCB on their personalized reward, then their query points would quickly concentrate around  $x_i^*$  making it difficult for a client  $j$  to satisfactorily learn  $h_i$  (and hence  $f_j$ ) especially in regions far away from  $x_i^*$ , even if all the observations of client  $i$  were available to client  $j$ . Hence, Collaborative Exploration is necessary for each client to learn the maximizer  $x_i^*$  of its personalized objective function  $f_i$ . This adds another layer of nuance to the exploration-exploitation trade-off as the exploration at any client needs to serve the goal of greater good instead of being sufficient for local exploitation.

Inspired by *deterministic sequencing of exploration and exploitation (DSEE)* algorithm for discrete bandits for centralized learning [302], we propose CEPE, a

kernel-based collaborative learning policy that effectively trades off exploration and exploitation in a distributed learning setting. We first describe the design of the interleaving exploration and exploitation epochs, a central component in CEPE. Following that, we describe the communication protocol and the query policy adopted in CEPE that ensures a low communication cost along with an overall low regret.

### The interleaving epoch structure

CEPE divides the time horizon of  $T$  steps into interleaving epochs of exploration and exploitation of increasing lengths. The length of each epoch is fixed at the beginning of the algorithm based on  $N_t : \mathbb{N} \rightarrow \mathbb{R}$ , a positive non-decreasing sequence of real numbers that is assumed to have been provided as an input to CEPE. Let  $\mathcal{A}(t)$  denote the collection of time instants at which CEPE had carried out an exploration step up to time  $t-1$ . CEPE decides to explore at a time instant  $t$  if  $|\mathcal{A}(t)| \leq N_t$ , otherwise it decides to exploit at time  $t$ . As a concrete example, consider the case where  $N_t = t^{2/3}$  for  $t \in N$ . At  $t = 1$ ,  $|\mathcal{A}(1)| = 0$  as no point has been explored before it. Since  $N_1 = 1 > |\mathcal{A}(1)|$ , CEPE explores at  $t = 1$ . For  $t = 2$ , evidently  $|\mathcal{A}(2)| = 1$  and  $N_2 = 2^{2/3} > 1$  implying CEPE again explores at  $t = 2$ . Similarly, CEPE also explores at  $t = 3$  as  $|\mathcal{A}(3)| = 2 < N_3$ . However at  $t = 4$ ,  $|\mathcal{A}(4)| = 3$  while  $N_4 < 3$ . Consequently, CEPE begins an exploitative epoch at  $t = 4$  and extends to  $t = 5$  as  $|\mathcal{A}(5)| = 3 > N_5$ . Once again at  $t = 6$ ,  $|\mathcal{A}(6)| < N_6$  and CEPE decides to explore, terminating the first exploitative epoch and beginning the second exploratory epoch. This process repeats until the end of the time horizon.

The design of the exploration-exploitation epochs is a central piece of the

puzzle as it is related to both the communication cost and the regret due to its implicit links with the communication protocol of CEPE and the design of the update rule of the decision rule, as described below.

## Communication Protocol

CEPE adopts a straightforward, easy to implement communication and message exchange protocol. All the communication between the clients and the server happens only during an exploration epoch. CEPE forgoes communication during exploitation epoch to ensure a low communication overhead. Thus, the separation between exploration and exploitation allows effective control of information exchange across clients for learning the global objective while simultaneously helping limit the communication overhead.

At every time instant  $t$  during an exploration epoch, each client sends the value of the current decision variable queried,  $x_{i,t}$  along with the observed random reward,  $y_{i,t}$  to the server. The server then broadcasts this to all clients.

We would like to point out that such a communication scheme does not violate any privacy concerns and is designed in a similar spirit to various distributed learning algorithms, especially in the scenario of Federated Learning. Similar to any FL algorithm, CEPE only communicates the current model parameters (the decision variable  $x$  in this setup) and not the actual data. The only difference is that CEPE also communicates a random loss associated with the current decision variable, which also does not give access to any actual data held by the clients. Thus, CEPE also ensures privacy by avoiding data sharing as considered in FL setups.

## Query Policy

The only step left to specify is the update rule of decision variable,  $x$ , used in CEPE. The update rule is based on whether the current time instant belongs to an explorative or an exploitative epoch and is guided by a fictitious Gaussian process (GP) prior for the elements of the RKHS, as described in Section 6.2.2. During an exploratory epoch, the clients choose different points across the epoch for continual learning. In particular, client  $i$  chooses  $x_{i,t} = \arg \max_{x \in \mathcal{X}} \sigma_{t-1}^{(h_i)}(x)$ , where  $\sigma_{t-1}^{(h_i)}(\cdot)$  is the posterior standard deviation of the GP model of  $h_i$  based on all the previous exploratory observations  $\mathcal{S}_{i,t} = \{(x_{i,s}, y_{i,s}) : s \in \mathcal{A}(t)\}$  of client  $i$  according to Eqn. (6.5). On the other hand, during an exploitation epoch the query point is fixed and chosen to maximize earning and forgo learning. Specifically, each client  $i$  chooses a point with the highest predicted value based on the GP model, i.e.,  $x_{i,t} = \arg \max_{x \in \mathcal{X}} \mu_{t-1}^{(f_i)}(x)$ , where  $\mu_{t-1}^{(f_i)}(\cdot)$  is the posterior mean of the personalized reward function, given as

$$\mu_{t-1}^{(f_i)}(\cdot) = \alpha_i \mu_{t-1}^{(h_i)}(\cdot) + \frac{(1 - \alpha_i)}{K} \sum_{j=1}^K \mu_{t-1}^{(h_j)}(\cdot).$$

In the above expression  $\mu_{t-1}^{(h_j)}(\cdot)$  are the posterior means of the GP model of  $h_j$  based on all the previous exploratory observations  $\mathcal{S}_{j,t}$  corresponding to client  $j = 1, 2, \dots, K$ . We also provide a pseudocode in Alg. 15 that succinctly combines the three design components of CEPE.

### 6.3.2 Performance Analysis

The theorem below establishes an upper bound on the regret performance of CEPE.

---

**Algorithm 15** Collaborative Exploration with Personalized Exploitation (CEPE)

---

**Input:**  $\{B_j\}_{j=1}^K$ , the kernel  $k(\cdot, \cdot)$ ,  $\{N_t\}_{t \in \mathbb{N}}$   
Set  $t \leftarrow 1$ ,  $\mathcal{A}(1) \leftarrow \emptyset$   
**repeat**  
    **if**  $|\mathcal{A}(t)| < N_t$  **then**  
         $\mathcal{A}(t+1) = \mathcal{A}(t) \cup \{t\}$   
    **end if**  
    **if**  $t \in \mathcal{A}(t+1)$  **then**  
         $x_{i,t} = \arg \max_{x \in \mathcal{X}} \sigma_{t-1}^{(h_i)}(x)$   
    **else**  
         $x_{i,t} = \arg \max_{x \in \mathcal{X}} \mu_{t-1}^{(f_i)}(x)$   
    **end if**  
    Query the function  $h_i$  at  $x_{i,t}$  to obtain  $y_{i,t}$   
    **if**  $t \in \mathcal{A}(t+1)$  **then**  
        Send  $(x_{i,t}, y_{i,t})$  to the server which is then broadcast to all clients  
    **end if**  
     $t \leftarrow t + 1$   
**until**  $t = T$

---

**Theorem 6.3.1.** Consider the kernel-based collaborative learning with personalized rewards setting described in Section 6.2.1. Under Assumptions 6.2.1 and 6.2.2 and for a given sequence  $\{N_t\}_{t \in \mathbb{N}}$ , the regret performance of CEPE satisfies, for any  $\delta_0 \in (0, 1)$ , with probability at least  $1 - \delta_0$ ,

$$R_{CEPE}(T) = O\left(KN_T + KT \sqrt{\frac{\gamma_{N_T}}{N_T} \log\left(\frac{T}{\delta_0}\right)}\right)$$

In particular, the regret is minimized with the choice of  $N_T = \Theta(T^{\frac{2}{3-\kappa}} (\log(T/\delta_0))^{\frac{1}{3}})$  in which case,

$$R_{CEPE}(T) = O(KT^{\frac{2}{3-\kappa}} (\log(T/\delta_0))^{\frac{1}{3}}).$$

Recall that  $\gamma_t = O(t^\kappa)$  with  $\kappa \in (0, 1]$  corresponds to the maximal information gain of the kernel  $k$ . Specifically, if the underlying kernel is a Matérn kernel with smoothness parameter  $\nu$ , then it is known that  $\kappa = d/(2\nu + d)$  [301]. Consequently, the regret incurred by CEPE is given as  $O(T^{\frac{2\nu+d}{3\nu+d}})$ . In the case of a Squared Exponential kernel,  $\kappa \rightarrow 0$  and hence  $R_{CEPE}(T) = O(T^{2/3} (\log(T))^{d/6})$ .

*Proof.* We here provide a sketch of the proof. We bound the regret in the exploration and exploitation epochs separately. The bound on the RKHS norm of  $f_i$  implies a bound on its sup norm. Thus, the regret in the exploration epoch is simply bounded by  $O(KN_T)$ . The regret during the exploitation epoch is bounded using Lemma 6.2.4 based on the posterior variance of the GP model. Maximal uncertainty reduction sampling and a known bound on the total uncertainty (cumulative conditional variances) of a GP model based on information gain allows us to bound the posterior variance with an  $\mathcal{O}(\frac{\gamma_{N_t}}{N_t})$  term. Combining these two results, we bound the regret in the exploitation epoch. A detailed proof of Theorem 6.3.1 is provided in Appendix E.1.  $\square$

*Remark 6.3.2.* The upper bound on the regret of CEPE given in Theorem 6.3.1 is sublinear in  $T$  as  $\gamma_T = o(T)$  for typical kernels [301]. A sublinear regret bound guarantees the convergence to the maximum personalized reward  $f_i(x_i^*)$  across all clients, as  $T$  grows. We would like to emphasize the significance of this guarantee for the performance of CEPE. In comparison, it is not clear whether standard algorithms such as GP-UCB or GP-TS achieve sublinear regret, even in the simpler problem of a single client. The existing upper bounds on the regret performance of these algorithms are in the form of  $\tilde{O}(\gamma_T \sqrt{T})$ , which may be trivial, as  $\gamma_T$  may grow faster than  $\sqrt{T}$ . For example, that is the case with a broad range of parameters in the case of Matérn kernel (see [305] for a detailed discussion). The CEPE algorithm thus may be of interest even for a single client setting, in terms of introducing a simple algorithm with sublinear regret. We, however, note that more sophisticated algorithms such as GP-ThreDS [247], SupKernelUCB [308], and BPE [193] achieve better regret bounds in the case of a single client.

We would like to emphasize the role of the sequence  $\{N_t\}_{t \in \mathbb{N}}$  in the context of

the performance of CEPE, both in terms of regret and communication cost. It affects both these metrics through the overall amount of exploration carried out in CEPE. In particular, at the end of the time horizon, CEPE would have explored for no more than  $N_T$  time steps. The impact of this on the regret can be noted from the statement of Theorem 6.3.1. The first term,  $O(KN_T)$ , corresponds to the regret incurred during the exploration sequence, and increases with  $N_T$ . The second term,  $O\left(KT \sqrt{\frac{\gamma_{N_T}}{N_T} \log(\frac{1}{\delta_0})}\right)$ , corresponds to the regret incurred during exploitation, and decreases with  $N_T$ . That is, a larger exploration sequence enables a better approximation of the arm statistics, which leads to a better performance during the exploitation sequence. On the other hand, since CEPE communicates at all and only the exploration time instants, its communication cost is bounded by  $N_T$ . Thus, CEPE controls the communication-regret trade-off through this sequence  $N_t$  and one can appropriately choose the sequence to trade-off the regret against communication based on the demands of a particular application. For example, if the objective is to minimize regret, then the optimal size  $N_T$  is obtained by minimizing the larger of the two terms in expression of regret, yielding us  $N_T = \Theta(T^{\frac{2}{3-\kappa}} (\log(1/\delta_0))^{\frac{1}{3}})$ , as referred to in Theorem 6.3.1. Under this scenario, CEPE incurs a communication cost of  $O(T^{2/(3-\kappa)})$ , which is sublinear in  $T$ .

## 6.4 Reducing Communication Cost via Sparse Approximation

In this section, we develop a variant of the CEPE algorithm that leverages sparse approximations of the posterior GP models to achieve significant reduction in communication cost while preserving the optimal regret performance. We begin with some preliminaries on sparse approximations of GP followed by the

description of our proposed algorithm.

### 6.4.1 Sparse Approximation of GP models

The sparse approximations of GP models are designed to approximate the posterior mean and variance obtained from the GP model, using a subset of query points to reduce the computational cost associated with evaluating the exact expressions. Let  $\mathbf{z}_t = \{z_1, z_2, \dots, z_m\} \subset \mathbf{x}_t$  be a subset of the query points, which we aim to use in approximating the posterior distribution of the GP model. These points are often referred to as the *inducing points*. The Nyström approximations are given as follows [327]:

$$\tilde{\mu}_t(x) = k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} \mathbf{y}_t \quad (6.6)$$

$$\begin{aligned} \tilde{\sigma}_t^2(x) &= \frac{1}{\lambda} (k(x, x) - k_{\mathbf{z}_t, x}^\top K_{\mathbf{z}_t, \mathbf{z}_t}^{-1} k_{\mathbf{z}_t, x} + \\ &\quad k_{\mathbf{z}_t, x}^\top (K_{\mathbf{z}_t, \mathbf{z}_t} + \lambda^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} k_{\mathbf{z}_t, x}) \end{aligned} \quad (6.7)$$

where  $K_{\mathbf{z}_t, \mathbf{z}_t} = [k(z_i, z_j)]_{i,j=1}^m \in \mathbb{R}^{m \times m}$ ,  $K_{\mathbf{z}_t, \mathbf{x}_t} = [k(z_i, x_j)]_{i=1, j=1}^{m, t} \in \mathbb{R}^{m \times t}$  and  $K_{\mathbf{x}_t, \mathbf{z}_t} = K_{\mathbf{z}_t, \mathbf{x}_t}^\top$ .

### 6.4.2 CEPE with Sparse Approximation

Referred to as S-CEPE, this variant of CEPE adopts a simpler design of interleaved exploration-exploitation phases. It consists of a single exploration phase of length  $N_T$ , entirely concentrated at the beginning of the time horizon, followed by an exploitation phase till the end of the time horizon. Such a design provides the clients with the entire exploration sequence which is required to construct the set of inducing points in Nyström approximation.

Unlike CEPE, the communication in S-CEPE is completely carried out in the exploitation phase. In particular, all the communication is carried during the first  $N_T^{(c)}$  steps of the exploitation phase, which are referred to the communication phase. At the  $s^{\text{th}}$  instant during this communication phase, client  $i$  sends  $(z_{i,s}, \mathbf{w}_{i,s})$  to the server, where  $z_{i,s}$  denotes the  $s^{\text{th}}$  inducing point and  $\mathbf{w}_{i,s} \in \mathbb{R}$  denotes the  $s^{\text{th}}$  element of the vector  $\mathbf{w}_i$ . The construction of the inducing points and the vector  $\mathbf{w}_i$  is described below. The server then broadcasts this to all the clients. The length of the communication sequence,  $N_T^{(c)}$ , is set to  $9(1 + 1/\lambda)q_0\gamma_{N_T}$  to allow all clients to complete their communication with high probability (See Lemma 6.4.1), where  $q_0$  is a constant specified later.

The messages communicated during S-CEPE are determined using the sparse approximation of GP models. Specifically, at the end of the exploration epoch, each client  $i$  constructs its set of inducing points  $\mathbf{z}_{i,N_T} = (z_{i,1}, z_{i,2}, \dots, z_{i,M_i})$  by including each point  $x_{i,j} \in \mathbf{x}_{i,N_T}$  queried during the exploration epoch with a probability equal to  $q_0 [\sigma_{N_T}^{(h_i)}(x_{i,j})]^2$ , independent of other points. Here  $q_0$  is a constant that is used to appropriately scale the probabilities of inclusion to ensure a sufficiently large set for a good approximation [49] and  $M_i$  denotes the random cardinality of the set of inducing points of client  $i$ . Each client  $i$ , using their set of inducing points, then proceeds to compute the vector  $\mathbf{w}_i = (\lambda K_{\mathbf{z}_{i,N_T}, \mathbf{z}_{i,N_T}} + K_{\mathbf{z}_{i,N_T}, \mathbf{x}_{i,N_T}} K_{\mathbf{x}_{i,N_T}, \mathbf{z}_{i,N_T}})^{-1} K_{\mathbf{z}_{i,N_T}, \mathbf{x}_{i,N_T}} \mathbf{y}_{i,N_T}$ .

The query policy used for S-CEPE during the exploration phase is same as that for CEPE, that is, to query points based on maximum uncertainty. During the exploitation phase of S-CEPE, each client  $i$  chooses to query the maximizer of the predictive mean of their *local* observation function,  $\arg \max_{x \in \mathcal{X}} \mu_{N_T}^{(h_i)}(x)$  for the communication phase. After the communication phase, the clients query the

maximizer of the Nyström approximate posterior mean of their personalized reward function for the rest of the epoch. The approximate posterior mean of  $f_i$  is given as

$$\tilde{\mu}_{N_T}^{(f_i)}(\cdot) = \alpha_i \tilde{\mu}_{N_T}^{(h_i)}(\cdot) + \frac{(1 - \alpha_i)}{K} \sum_{j=1}^K \tilde{\mu}_{N_T}^{(h_j)}(\cdot)$$

with  $\tilde{\mu}_{N_T}^{(h_i)}$ 's as defined in Eqn. (6.6). A pseudocode for S-CEPE is provided in Algorithm 16.

### 6.4.3 Performance Analysis

We first establish a bound on the length of the communication sequence. The following lemma, which is a slightly modified version of Theorem 1 in [49], gives us an upper bound on the largest inducing set among all the clients.

**Lemma 6.4.1.** *Fix  $\delta \in (0, 1)$ ,  $\varepsilon \in (0, 1)$  and let  $\chi = \frac{1+\varepsilon}{1-\varepsilon}$ . If the set of inducing points is constructed as outlined in Section 6.4.2 with  $q_0 \geq 6\chi \log(4TK/\delta)/\varepsilon^2$ , then with probability at least  $1 - \delta$ ,  $\max_i M_i \leq 9(1 + 1/\lambda)q_0\gamma_{N_T}$ .*

We can conclude from the above lemma that choice of  $N_T^{(c)}$  used in S-CEPE allows for sufficient time for all the clients to transmit their messages to the server in the communication phase. As a result, the communication cost of S-CEPE can be bounded by  $N_T^{(c)} = O(\gamma_{N_T}) = O(N_T^K)$ . The communication cost of S-CEPE is significantly lower than that of CEPE. For example in the case of SE kernel with the optimal choice of  $N_T = O(KT^{\frac{2}{3}}(\log(T/\delta_0))^{(d+2)/6})$ , the communication cost of S-CEPE is bounded by  $O(K \log^{d+1}(T))$ , while the communication cost of CEPE is in  $O(KT^{\frac{2}{3}}(\log(T/\delta_0))^{(d+2)/6})$ . Furthermore, S-CEPE achieves this significant reduction in communication cost while maintaining the regret guarantees of CEPE, as shown in the following theorem.

**Theorem 6.4.2.** Consider the kernel-based collaborative learning with personalized rewards as described in Section 6.2.1. Under Assumptions 6.2.1 and 6.2.2, the regret performance of S-CEPE satisfies

$$R_{S\text{-CEPE}}(T) = O\left(KN_T + KT \sqrt{\frac{\gamma_{N_T}}{N_T} \log(T/\delta_0)}\right).$$

with probability at least  $1 - \delta_0$  for any  $\delta_0 \in (0, 1)$ .

*Proof.* The proof of this theorem is similar to that of Theorem 6.3.1 and we provide a sketch of the proof below. We bound the regret in exploration, communication and the exploitation phases separately. The regret during the exploration phase and the communication phase is bounded within a constant factor of their length, i.e.,  $O(KN_T)$  and  $O(K\gamma_{N_T})$  respectively. The exploitation regret is bounded following similar steps as in the proof of 6.3.1, except that we bound the instantaneous regret based on approximate posterior standard deviation in contrast to the exact one using Lemma 6.4.3, described below.  $\square$

The following lemma is a counterpart to Lemma 6.2.4, in the case of sparse approximation of the posterior mean and variance.

**Lemma 6.4.3.** Under Assumptions 6.2.1 and 6.2.2, provided a set of observations  $\mathcal{H}_t = \{\mathbf{x}_t, \mathbf{y}_t\}$ , with  $\mathbf{x}_t$  chosen independently of the associated noise sequence, and corresponding subset of inducing points  $\mathbf{z}_t$  obtained as outlined in Section 6.4.2, we have, for a fixed  $x \in \mathcal{X}$ , with probability at least  $1 - \delta$ ,

$$|h(x) - \tilde{\mu}_t(x)| \leq \tilde{\beta}(B, \delta) \tilde{\sigma}_t(x),$$

where  $\tilde{\mu}(x)$  and  $\tilde{\sigma}(x)$  are as defined in (6.7) and  $\tilde{\beta}(B, \delta) = B \sqrt{2\lambda/(1-\varepsilon)} + R \sqrt{2 \log(T/\delta)}$ .

Detailed proofs of Theorem 6.4.2 and Lemma 6.4.3 are provided in Appendix E.1.

---

**Algorithm 16 S-CEPE**


---

**Input:**  $N_T$ , the kernel  $k(\cdot, \cdot)$ ,  $\varepsilon \in (0, 1)$ ,  $\delta_0 \in (0, 1)$ ,  $\lambda$ .  
 $\mathbf{z}_{i,N_T} \leftarrow \{\}$ ,  $q_0 \leftarrow 6(1 + \varepsilon) \log(8TK/\delta_0)/\varepsilon^2(1 - \varepsilon)$ ,  $N_T^{(c)} \leftarrow 9(1 + 1/\lambda)q_0\gamma_{N_T}$

**for**  $t = 1, 2, \dots, N_T$  **do**

- Choose  $x_{i,t} = \arg \max_{x \in \mathcal{X}} \sigma_{t-1}^{(h_i)}(x)$
- Query the function  $h_i$  at  $x_{i,t}$  to obtain  $y_{i,t}$
- end for**
- for**  $j = 1, 2, \dots, N_T$  **do**

  - Draw  $W_j \sim \text{Bern}(p_j)$ , where  $p_j := q_0[\sigma_{N_T}^{(h_i)}(x_{i,j})]^2$
  - if**  $W_j = 1$  **then**

    - $\mathbf{z}_{i,N_T} \leftarrow \mathbf{z}_{i,N_T} \cup \{x_{i,j}\}$

  - end if**

- end for**
- Evaluate the vector  $\mathbf{w}_i = (\lambda K_{\mathbf{z}_{i,N_T}, \mathbf{z}_{i,N_T}} + K_{\mathbf{z}_{i,N_T}, \mathbf{x}_{i,N_T}} K_{\mathbf{x}_{i,N_T}, \mathbf{z}_{i,N_T}})^{-1} K_{\mathbf{z}_{i,N_T}, \mathbf{x}_{i,N_T}} \mathbf{y}_{i,N_T}$
- for**  $s = 1, 2, \dots, N_T^{(c)}$  **do**

  - $t \leftarrow t + 1$
  - $x_{i,t} \leftarrow \arg \max_{x \in \mathcal{X}} \mu_{N_T}^{(h_i)}(x)$
  - if**  $s \leq m_i$  **then**

    - Send  $(z_{i,s}, \mathbf{w}_{i,s})$  to the server which is then broadcast to all clients

  - end if**

- end for**
- repeat**

  - $t \leftarrow t + 1$
  - $x_{i,t} \leftarrow \arg \max_{x \in \mathcal{X}} \tilde{\mu}_{N_T}^{(f_i)}(x)$

- until**  $t = T$

---

## 6.5 Regret Lower Bound

In this section, we establish a lower bound on regret incurred by any algorithm for the problem of kernel-based collaborative learning with personalized rewards. In particular, in the following theorem, we show that even when there is no communication constraint, the total regret incurred by any learning algorithm as defined in Eqn. (6.3) is at least  $\Omega(T^{2/(3-\kappa)})$ , where  $\kappa$  characterizes the order of maximal information gain and depends on the underlying kernel.

**Theorem 6.5.1.** *Consider the kernel-based collaborative learning with personalized rewards setting described in Section 6.2.1, in an RKHS corresponding to the family of*

*Squared Exponential (SE) and Matérn (with smoothness  $\nu$ ) kernels with no constraints on communication. Then, for all sets of personalization parameters  $\{\alpha_1, \alpha_2, \dots, \alpha_K\}$ , there exists a choice of observation functions  $\{h_1, h_2, \dots, h_K\}$  for which, the cumulative regret of any policy  $\pi$ , satisfies*

$$\mathbb{E}[R_\pi(T)] = \begin{cases} \Omega(\alpha_* T^{\frac{2\nu+d}{3\nu+d}}) & \text{for Matérn kernels,} \\ \Omega(\alpha_* T^{2/3} (\log T)^{d/6}) & \text{for SE kernel.} \end{cases}$$

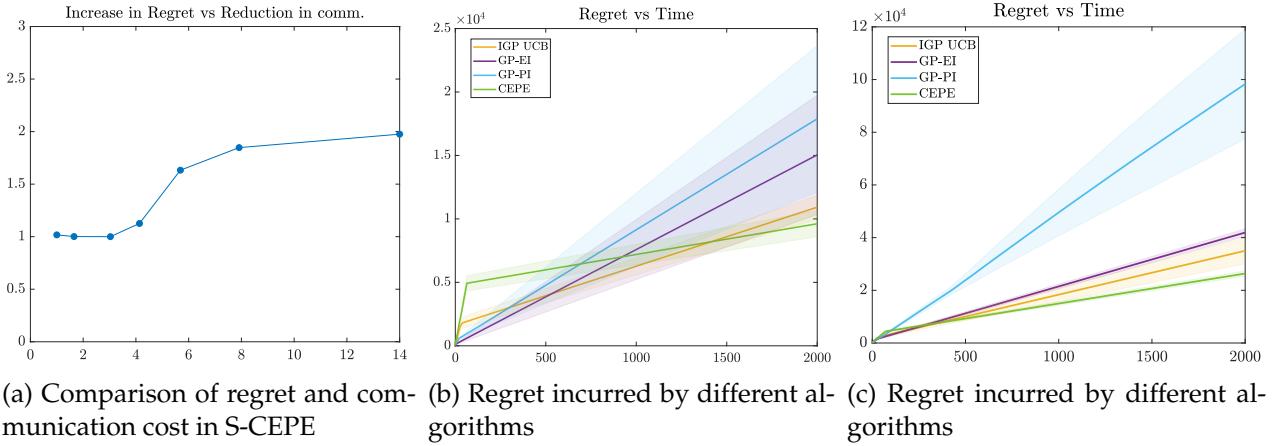
*for sufficiently large time horizon  $T$ . In the above expression,  $\alpha_* = \max_i(\min\{\alpha_i, 1 - \alpha_i\}) > 0$  is a constant that depends only on the personalization parameters.*

Theorem 6.5.1 shows that the regret performance of CEPE (and S-CEPE) is order optimal up to logarithmic factors. Please refer to Appendix E.3 for a detailed proof.

*Remark 6.5.2.* Shi *et al.* [270] also conjectured a lower bound for the regret in the finite arm setting. Their lower bound is, however, different in the sense that it is a distribution-dependent lower bound. The distribution-dependent lower bounds typically are given in terms of the gaps (in the mean and KL-divergence) between the best arms and suboptimal arms. These bounds do not apply to our continuum arm setting as such gaps are always zero in case of infinite arms. We thus use a different method to prove a minimax bound on the regret.

## 6.6 Empirical Studies

In this section, we corroborate our theoretical results with empirical evidence that compares the regret incurred by CEPE against several baseline algorithms and documents the reduction in communication cost offered by S-CEPE. We



(a) Comparison of regret and communication cost in S-CEPE      (b) Regret incurred by different algorithms      (c) Regret incurred by different algorithms

Figure 6.1: (a) Cumulative regret of S-CEPE after  $T = 2000$  steps relative to CEPE plotted against the reduction in communication offered relative to CEPE when S-CEPE is run with different sizes of inducing sets. (b, c) Cumulative regret incurred by different algorithms over  $T = 2000$  steps for different objective functions ((b) Branin, (c) Function in [274])

begin with describing the experimental setup followed by discussing the particular experiments and their results.

### 6.6.1 Experimental Setup

We consider a collaborative learning setup among  $K = 50$  clients. The personalization parameter of each client is drawn uniformly from the interval  $[0.1, 0.9]$ , independent of all other clients. The observation function for each client is drawn uniformly at random from the following set of 9 functions:  $\{h : h(x_1, x_2) = \Lambda(x_1^i, x_2^j), i, j \in \{1, 2, 3\}\}$ , where  $\Lambda : [0, 1]^2 \rightarrow \mathbb{R}$  is a standard two-dimensional benchmark function for Bayesian Optimization. We perform experiments for two different choices of  $\Lambda$ . The first such function is Branin, which is a standard benchmark function for Bayesian Optimization and its ana-

lytical expression is given below [20, 225]

$$B(x, y) = -\frac{1}{51.95} \left( \left( v - \frac{5.1u^2}{4\pi^2} + \frac{5u}{\pi} - 6 \right)^2 + \left( 10 - \frac{10}{8\pi} \right) \cos(u) - 44.81 \right), \quad (6.8)$$

where  $u = 15x - 5$  and  $v = 15y$ . The second one is the function from [274] which is given by

$$F(x, y) = (6x - 2)^2 \sin(12x - 4) + (6y - 2)^2 \sin(12y - 4). \quad (6.9)$$

We implement all algorithms using a uniform discretization of 900 points over the domain,  $[0, 1]^2$ . We use a SE kernel with lengthscale parameter 0.2. The observations are corrupted with Gaussian noise with  $\sigma^2 = 0.01$ , and  $\lambda$  is set to 0.01. The results in all the experiments are obtained for a time horizon of  $T = 2000$  steps by averaging over 5 Monte Carlo runs. We describe below the implementation details of IGP-UCB, PI and EI.

1. IGP-UCB: The algorithm is implemented exactly as outlined in [68] with  $B$  (in scaling parameter  $\beta_t$ ) set to 15 for both algorithms. The parameters  $R$  and  $\delta_0$  are set to  $10^{-2}$  and  $10^{-3}$ .  $\gamma_t$  is set to  $\log t$ .
2. Expected Improvement(EI)/Probability of Improvement (PI): Similar to IGP-UCB, EI and PI select the observation points based on maximizing an index often referred to as an acquisition function. The acquisition function of EI is  $(\mu(x) - f^* - \varepsilon)\Phi(z) + \sigma(x)\phi(z)$ , where  $z = \frac{\mu(x)-f^*-\varepsilon}{\sigma(x)}$ . The acquisition function of PI is  $\Phi(z)$ , where  $z = \frac{\mu(x)-f^*-\xi}{\sigma(x)}$ . Here,  $\Phi(\cdot)$  and  $\phi(\cdot)$  denote the CDF and PDF of a standard normal random variable.  $f^*$  is set to be the maximum value of  $\mu(x)$  among the current observation points. The parameters  $\varepsilon$  and  $\xi$  are used to balance the exploration-exploitation trade-off. We follow [137] that showed the best choice of these parameters are small non-zero values. In particular,  $\varepsilon$  and  $\xi$  are both set to 0.01.

## 6.6.2 Results

In the first experiment, we compare the regret performance of CEPE against baseline algorithms: IGP-UCB [68], GP-EI (Expected Improvement) [322] and GP-PI (Probability of Improvement) [325]. Since these baseline algorithms are designed for the single client setting, we appropriately modify them to adapt them to the collaborative learning setup considered in this work. Specifically, for the baseline algorithms we consider the benign setting where at each time instant  $t$ , the entire history of all the query points and observations of each client upto time  $t - 1$  is accessible by all other clients. Each client  $i$  uses this history of observations upto time  $t - 1$  to construct the estimate of the observation functions of other clients which in turn is used to construct the acquisition function for the query point at time instant  $t$  to maximize its personalized objective function,  $f_i$ . The major difference here is that query points are chosen *greedily* by the clients to maximize their own objective function which might make it difficult for other clients to satisfactorily estimate (and optimize) their own objective functions.

We perform this comparison for two different choices of observation functions. In the first case, the base function  $\Lambda$  is set to the Branin function (See Eqn. (6.8)) and the overall cumulative regret incurred by different algorithms is plotted in Fig. 6.1b. In the second case,  $\Lambda$  is set to the two-dimensional extension of the benchmark function proposed in [274] (See Eqn. (6.9)) and overall the cumulative regret for different algorithms for this experiment is plotted in Fig. 6.1c. As it can be seen from both the plots, the regret incurred by CEPE is lesser than that incurred by other baseline algorithms. The results are consistent with our discussion in Section 6.3.1 showing that baseline GP bandit algorithms do not perform well in the collaborative setting, even with communication at

each round. The altruistic observations obtained during the exploration sequence in CEPE helps all the clients and contribute towards an overall lower cumulative regret.

In the second experiment, we study the trade-off between communication cost and regret offered by S-CEPE. In particular, we evaluate the overall cumulative regret of S-CEPE for varying sizes of the set of inducing points obtained by varying  $q_0$ . Recall that the number of inducing points determines the communication cost. This is compared against the regret and communication cost incurred by CEPE. For both CEPE and S-CEPE, we set  $N_T = 64$  which also determines the communication cost of CEPE. The objective functions for this case are obtained by setting  $\Lambda$  to the Branin function. We plot the ratio of overall cumulative regret between S-CEPE and CEPE against the ratio between the communication cost of CEPE to S-CEPE in Fig. 6.1a. As seen from the plot, S-CEPE offers upto a 14-fold reduction in communication cost while sacrificing regret by a factor of less than 2. This shows the efficiency of sparse approximation methods in practice.

## **Part III**

# **Active Learning and Active Hypothesis Testing**

## CHAPTER 7

### ONLINE ACTIVE LEARNING FOR LABEL EFFICIENCY

#### 7.1 Introduction

We consider online classification of streaming instances within the framework of statistical learning theory. Let  $\{X_t\}_{t \geq 1}$  be a sequence of instances drawn independently at random from an unknown underlying distribution  $\mathbb{P}_X$  over an instance space  $\mathcal{X}$ . Each instance  $X_t$  has a hidden binary label  $Y_t \in \{0, 1\}$  that relates probabilistically to the instance according to an unknown conditional distribution  $\mathbb{P}_{Y|X}$ . The learner is characterized by its hypothesis space  $\mathcal{H}$  consisting of all classifiers under consideration. At each time  $t$ , the learner decides whether to query the label of the current instance  $X_t$ . If yes,  $Y_t$  is revealed. Otherwise, the learner predicts the label of  $X_t$  using a hypothesis in  $\mathcal{H}$  and incurs a classification error if the predicted label does not equal to the true label  $Y_t$ . The objective is to minimize the expected number of queries over a horizon of length  $T$  while constraining the total number of classification errors. The tension between label complexity and classification error rate needs to be carefully balanced through a sequential strategy governing the query and labeling decisions at each time.

The above problem arises in applications such as spam detection and event detection in real-time surveillance. The key characteristics of these applications are the high-volume streaming of instances and the complex and nuanced definition of labels. While the latter necessitates human intervention to provide annotations for selected instances, such human annotations, time consuming and expensive to obtain, should be sought after sparingly to ensure scalability.

### 7.1.1 Previous Work on Active Learning

The above problem falls under the general framework of active learning. In contrast to passive learning where labeled examples are given *a priori* or drawn at random, active learning asserts control over which labeled examples to learn from by actively querying the labels for carefully selected instances. The hope is that by learning from the most informative examples, the same level of classification accuracy can be achieved with much fewer labels than in passive learning.

**Offline Active Learning** Active learning has been studied extensively under the Probably Approximately Correct (PAC) model, where the objective is to output an  $\epsilon$ -optimal classifier with probability  $1 - \delta$  using as few labels as possible. The PAC model pertains to offline learning since the decision maker does not need to self label any instances during the learning process. An equivalent view is that classification errors that might have incurred during the learning process are inconsequential, and the tension between label complexity and classification errors is absent. If measured purely by label complexity, the decision maker has the luxury of skipping, at no cost, as many instances as needed to wait for the most informative instance to emerge. In the online setting considered in this work, however, self labeling is required in the event of no query, classification errors need to be strictly constrained, and no feedback to the predicted labels is available (thus learning has to rely solely on queried labels). As a result, online active learning faces an essential tradeoff between real-time classification errors and label complexity, which is absent in the offline settings.

A much celebrated offline active learning algorithm was given by Cohn, At-

las, and Ladner [70]. Named after its inventors, the CAL algorithm is applicable to a general hypothesis space  $\mathcal{H}$ . It, however, relies on the strong assumption of realizability, i.e., the instances are perfectly separable and there exists an error-free classifier in  $\mathcal{H}$ . In this case, hypotheses inconsistent with a single label can be safely eliminated from further consideration. Based on this key fact, CAL operates by maintaining two sets at each time: the *version space* consisting of all surviving hypotheses (i.e., those that are consistent with all past labels), and the *region of disagreement* (RoD), a subset of  $\mathcal{X}$  for which there is disagreement among hypotheses in the current version space regarding their labels. CAL queries labels if and only if the instance falls inside the current RoD. Each queried label reduces the version space, which in turn may shrink the RoD, and the algorithm iterates indefinitely. Note that instances outside the RoD are given the same label by all the hypotheses in the current version space. It is thus easy to see that CAL represents a conservative approach: it only disregards instances whose labels can already be perfectly inferred from past labels. Quite surprisingly, by merely avoiding querying labels that carry no additional information, exponential reduction in label complexity can be achieved in a broad class of problems. (See, for example, an excellent survey by Dasgupta [83] and a monograph by Hanneke [128]).

The CAL algorithm was extended to the agnostic setting by Balcan, Beygelzimer, and Langford [21]. In the agnostic setting, instances are not separable, and even the best classifier  $h^*$  in  $\mathcal{H}$  experiences a non-zero error rate. The main challenge in extending CAL to the agnostic case is the update of the version space: a single inconsistent label can no longer disqualify a hypothesis, and the algorithm needs to balance the desire of quickly shrinking the version space with the irreversible risk of eliminating  $h^*$ . Referred to as  $A^2$  (Agnostic Ac-

tive), the algorithm developed by Balcan, Beygelzimer, and Langford explicitly maintains an  $\epsilon$  neighborhood of  $h^*$  in the version space by examining the empirical errors of each hypothesis. Analysis of the  $A^2$  algorithm can be found in [125, 126, 167, 127]. Variants of the  $A^2$  algorithm include [32, 34, 33, 84, 129]. In particular, the DHM algorithm (named after the authors) in [84] simplifies the maintenance of the RoD through a reduction to supervised learning.

The above conservative approach originated from the CAL algorithm is referred to as the disagreement-based approach. The design methodology of this conservative approach focuses on avoiding querying labels that provide no or little additional information. More aggressive approaches that actively seeking out more informative labels to query have been considered in the literature. One such approach is the so-called margin-based. It is specialized for learning homogeneous (i.e. through the origin) linear separators of instances on the unit sphere in  $\mathbb{R}^d$  and adopts a specific noise model that assumes linearity in terms of the inner product with the Bayes optimal classifier. In this case, the informativeness of a potential label can be measured by how close the instance is to the current decision boundary. Representative work on the margin-based approach includes [85, 22, 23, 19, 17, 18, 337, 73]. Another such approach is considered in Zhang and Chaudhari [338] that combines ideas from both disagreement-based and margin-based approaches.

Besides the stream-based model where instances arrive one at a time, active learning has also been considered under the synthesized instances and the pool-based sampling models [261] and synthesizes instances for various models for applications (see for example, [134, 296, 195]). These models are less relevant to the online setting considered in this work. In addition to online classification,

active learning approaches have also been considered in the context of Bayesian Learning [219, 292] and inference of network topology [255].

**Online Active Learning** Active learning in the online setting has received much less attention. The work of Cesa-Bianchi *et al.* [61] and Cavallanti *et al.* [59] extended the margin-based approach to the online setting, focusing, as in the offline case, on homogeneous linear separators for instances on the unit sphere in  $\mathbb{R}^d$ . A specific noise model was adopted, which assumes that the underlying conditional distribution of the labels is fully determined by the Bayes optimal classifier  $h^*$ . In this work, we consider a general instance space and arbitrary classifiers. Tackling the general setting, the proposed algorithm and the analysis are fundamentally different from these two existing studies. Furthermore, we show with simulation examples in Section 7.6 that, even when restricted to the special case of homogeneous linear separators, the algorithm proposed in this work outperforms the margin-based algorithm developed in [61, 59].

The only works we are aware of that extend the disagreement-based approach to the online setting are by Yang [333] and Cortes *et al.* [73]. Cortes *et al.* [73] constructs a disagreement graph to relate different hypotheses in the hypothesis space and then uses importance weighted sampling to eliminate them based on observation — an approach that is difficult to extend for the model considered in this work. Yang [333] extends the offline DHM algorithm to a stream-based setting and we discuss in detail the difference between [333] and this work in Section 7.1.2.

### 7.1.2 Main Results

We consider a general instance space  $\mathcal{X}$ , a general hypothesis space  $\mathcal{H}$  of Vapnik-Chervonenkis (VC) dimension  $d$ , and the Tsybakov noise model parameterized by  $\alpha \in (0, 1]$  [298]. We develop an online active learning algorithm and establish its  $O(dT^{\frac{2-2\alpha}{2-\alpha}} \log^2 T)$  label complexity and uniformly bounded regret in prediction errors with respect to the best classifier  $h^*$  in  $\mathcal{H}$ . More specifically, the total expected classification errors in excess to  $h^*$  over a horizon of length  $T$  is bounded below  $1/2$  independent of  $T$ , demonstrating that the proposed algorithm offers practically the same level of classification accuracy as  $h^*$  with a sublinear label complexity in  $T$ . We further establish a matching (up to a poly-logarithmic factor) lower bound, demonstrating the order optimality of the proposed algorithm. We address the tradeoff between label complexity and regret and show that the algorithm can be modified to operate at a different point on the tradeoff curve. Below we contextualize this work with respect to the existing literature by highlighting the differences in three aspects: algorithm design, analysis techniques, and performance comparison.

**Algorithm Design** Referred to as OLA (OnLine Active), the algorithm developed in this work is rooted in the design principle of the disagreement-based approach. The defining characteristic of the disagreement-based approach is to avoid querying instances that see insufficient disagreement among surviving hypotheses by maintaining, explicitly or inexplicitly, the RoD. Specific algorithm design differs in its temporal structure of when to update the RoD and, more crucially, in the threshold design on what constitutes sufficient disagreement. As detailed below, OLA differs from representative disagreement-based

algorithms—the offline  $A^2$  [21] and DHM [84] algorithms and the online ACAL algorithm [333]—in both aspects.

In terms of temporal structure, OLA operates in epochs and updates the RoD at the end of each epoch, where an epoch ends when a fixed number  $M$  of labels have been queried. This structure is different from  $A^2$ , DHM, and ACAL. In particular, the epochs in  $A^2$  are determined by the time instants when the size of the current RoD shrinks by half due to newly obtained labels. Such an epoch structure, however, requires the knowledge of the marginal distribution  $\mathbb{P}_X$  of the instances for evaluating the size of the RoD. The epoch structure of OLA obviates the need for this prior knowledge. DHM, on the other hand, does not operate in epochs and updates (inexplicably) the RoD at each time. Similarly, ACAL also updates the RoD at each time<sup>1</sup>. Moreover, the updates involve calculating thresholds by solving multiple non-convex optimization problems with randomized nonlinear constraints that can only be checked numerically. In contrast, the epoch-based updates in OLA only involve thresholds that are given in closed-form in terms of empirical errors.

A more crucial improvement in OLA is the design of the threshold that determines the RoD. This is the key algorithm parameter that directly controls the tradeoff between label complexity and classification error rate. By focusing only on empirical errors incurred over significant  $(X, Y)$  examples determined by the current RoD, we obtain a tighter concentration inequality and a more aggressive threshold design (See Theorem 7.4.1), which leads to significant reduction in label complexity as compared with  $A^2$ , DHM, and ACAL, as well as margin-based

---

<sup>1</sup>ACAL has a predetermined epoch structure with geometrically growing epoch length. This epoch structure, however, is not for controlling when to update the RoD, but rather for setting a diminishing sequence of outage probability of eliminating  $h^*$ . The algorithm otherwise restarts by forgetting all past experiences at the beginning of each epoch.

algorithms (see details on the performance comparison below).

**Analysis Techniques** Under the offline PAC setting, the label complexity of an algorithm is often analyzed in terms of the suboptimality gap  $\epsilon$  and the outage probability  $\delta$ . Under the online setting, however, the label complexity of an algorithm is measured in terms of the horizon length  $T$ , which counts both labeled and unlabeled instances. In the analysis of the label complexity of  $A^2$  [21, 125], unlabeled instances are assumed to be cost free, and bounds on the number of unlabeled instances skipped by the algorithm are missing and likely intractable. Without a bound on the unlabeled data usage, the offline label complexity in terms of  $(\epsilon, \delta)$  cannot be translated to its online counterpart.

Yang [333] analyze the label complexity by bounding the excess risk in terms of local Rademacher complexity [168] within each epoch. This technique is restricted to the specific threshold design in ACAL, which is based on expensive non-convex optimization with constraints on randomized Rademacher process.

We adopt new techniques in analyzing the online label complexity of OLA. First we separate the analysis into two stages based on the size of the RoD. For the early stage where the RoD is large, we show that RoD is decreasing exponentially. Then, to upper bound the label complexity, the key idea is to construct a supermartingale  $\{S(t)\}_{t \geq 0}$  given by the difference of an exponential function of the total queried labels up to  $t$  and a linear function of  $t$ . The optimal stopping theorem for supermartingales then leads to an upper bound on the exponential function of the label complexity. A bound on the label complexity thus follows from Jensen's inequality. The remaining label complexity where the RoD is small can be bounded by the product of the size and the remaining

time horizon. The separation of the two stages is then optimized to tighten the bound.

The lower bound established in this work is new. We are not aware of any existing lower bound on label complexity in the online setting. Lower bounds for the offline PAC setting (see, e.g., [58, 130]) are inapplicable to the online setting and were established using different techniques.

**Performance Comparison** We now comment on the performance comparison in terms of both asymptotic orders and finite-time performance.

As stated above, the performance analysis of  $A^2$  is in terms of the PAC parameters  $(\epsilon, \delta)$ . The analysis of its online performance is missing. Dasgupta, et. al provided an upper bound on the unlabeled data usage in DHM [84]. The bound, however, appears to be loose and translates to a linear  $O(T)$  label complexity in the online setting. Yang [333] provided an upper bound  $O(dT^{\frac{2-2\alpha}{2-\alpha}} \log^3 T)$  on the label complexity of ACAL, which is higher than the  $O(dT^{\frac{2-2\alpha}{2-\alpha}} \log^2 T)$  order offered by OLA.

The margin-based algorithm for learning homogeneous linear separators under a uniform distribution of  $X$  on the unit sphere is analyzed in [59] under the Tsybakov noise condition. It leads to a regret order of  $O(dT^{\frac{2-2\alpha}{3-2\alpha}} \log T)$  and a label complexity of  $O(dT^{\frac{2-2\alpha}{2-\alpha}} \log T)$  under the Tsybakov low noise condition. These orders cannot be directly compared with that of OLA due to the restrictions to homogeneous linear separators and the specific form of  $\mathbb{P}_{Y|X}$ . This margin-based algorithm also operates at a different point on the tradeoff curve between regret and label complexity, offering a slightly lower order in label complexity but a higher order in regret. However, even when restricted to the special case

targeted by this margin-based algorithm, the dominating polynomial term is the same, and the finite-time comparison given by simulation examples in Section 7.6 actually show superior performance of OLA in both label complexity and regret.

The finite-time comparison in Section 7.6 also demonstrate significant performance gain offered by OLA over the three representative disagreement-based algorithms:  $A^2$ , DHM, and ACAL. In particular, the improvement over the online algorithm ACAL is drastic.

## 7.2 Problem Formulation

### 7.2.1 Instances and Hypotheses

Let  $\{X_t\}_{t \geq 1}$  be a streaming sequence of instances, each drawn from an instance/sample space  $\mathcal{X}$  and characterized by its feature vector. Each subset of  $\mathcal{X}$  is a concept. There is a target concept  $C \subset \mathcal{X}$  that the learner aims to learn (e.g., learning the concept “table” from household objects). Relating to the target concept  $C$ , each instance  $X_t$  has a hidden label  $Y_t$ , indicating whether  $X_t \in C$  (i.e., a positive example wherein  $Y_t = 1$ ) or  $X_t \notin C$  (a negative example with  $Y_t = 0$ ). The label  $Y_t$  relates probabilistically to  $X_t$  according to an unknown conditional distribution  $\mathbb{P}_{Y|X}$ .

The learner is characterized by its hypothesis space  $\mathcal{H}$  consisting of all classifiers under consideration. Each hypothesis  $h \in \mathcal{H}$  is a measurable function mapping from  $\mathcal{X}$  to  $\{0, 1\}$ . The complexity of the hypothesis space  $\mathcal{H}$  is mea-

sured by its VC dimension  $d$ .

### 7.2.2 Error Rate, Disagreement, and Bayes Optimizer

Recall that  $\mathbb{P}_{Y|X}$  denotes the conditional distribution of the true label  $Y$  for a given  $X$ . Let  $\mathbb{P}_X$  denote the unknown marginal distribution of instances  $X$  and  $\mathbb{P} = \mathbb{P}_X \times \mathbb{P}_{Y|X}$  the joint distribution of an example  $(X, Y)$ . The error rate of a hypothesis  $h$  is given by

$$\epsilon_{\mathbb{P}}(h) = \mathbb{P}[h(X) \neq Y], \quad (7.1)$$

which is the probability that  $h$  misclassifies a random instance. Define the *pseudo-distance* and the *disagreement* between two hypotheses as, respectively,

$$d(h, h') = |\epsilon_{\mathbb{P}}(h) - \epsilon_{\mathbb{P}}(h')|, \quad \rho(h, h') = \mathbb{P}_X[h(X) \neq h'(X)], \quad (7.2)$$

where the distance is the difference in error rates and the disagreement is the probability mass of the instances over which the two hypotheses disagree. Lastly,  $\mathcal{D}(h_1, h_2) = \{x \in \mathcal{X} : h_1(x) \neq h_2(x)\}$  denotes the disagreement region between two hypotheses  $h_1$  and  $h_2$ .

Let  $h^*$  be the Bayes optimal classifier that minimizes the error rate, i.e., for all  $x \in \mathcal{X}$ ,  $h^*(x)$  is the label that minimizes the probability of classification error:

$$h^*(x) = \arg \min_{y=0,1} \mathbb{E}_{\mathbb{P}_{Y|X=x}} \mathbb{1}[Y \neq y], \quad (7.3)$$

where  $\mathbb{1}[\cdot]$  is the indicator function. Let

$$\eta(x) = \mathbb{P}_{Y|X=x}(Y = 1 | X = x). \quad (7.4)$$

It is easy to see that

$$h^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq \frac{1}{2} \\ 0 & \text{if } \eta(x) < \frac{1}{2} \end{cases}. \quad (7.5)$$

We assume that  $h^* \in \mathcal{H}$ .

### 7.2.3 Noise Condition

The function  $\eta(x)$  given in Eqn. (7.4) is a measure of the feature noise level at  $x$ . The noise-free case is when labels are deterministic:  $\mathbb{P}_{Y|X=x}$ , hence  $\eta(x)$ , assumes only values of 0 and 1. In this case, the optimal classifier  $h^*$  is error-free. This is referred to as the realizable case with perfectly separable data.

In a general agnostic case with arbitrary  $\mathbb{P}_{Y|X}$ , consistent classifiers may not exist, and even  $h^*$  suffers a positive error rate. A particular case, referred to as the Massart bounded noise condition [206], is when  $\eta(x)$  is discontinuous at the boundary between positive examples  $X_1^* \triangleq \{x \in \mathcal{X} : h^*(x) = 1\}$  and negative examples  $X_0^* \triangleq \{x \in \mathcal{X} : h^*(x) = 0\}$ . Specifically, there exists  $\gamma > 0$  such that  $|\eta(x) - \frac{1}{2}| \geq \gamma$  for all  $x \in \mathcal{X}$ .

A more general noise model is the Tsybakov noise condition [298], for which the Massart bounded noise condition is a special case. It allows  $\eta(x)$  to pass  $\frac{1}{2}$  with a continuous change across the decision boundary and parameterizes the slope around the boundary. Specifically, the Tsybakov noise condition states that there exist  $\alpha \in (0, 1]$ ,  $c_0 \geq 0$ , such that for all  $h$ , we have

$$\rho(h, h^*) \leq c_0 d^\alpha(h, h^*). \quad (7.6)$$

At  $\alpha = 1$ , the Tsybakov noise reduces to the more benign Massart noise. In terms of the slope around the decision boundary, the above condition can be restated as

$$\mathbb{P}_X \left( \left\{ x : \left| \eta(x) - \frac{1}{2} \right| \leq \gamma \right\} \right) \leq c'_0 \gamma^{\frac{\alpha}{1-\alpha}} \quad (7.7)$$

for some constant  $c'_0 \geq 0$ .

### 7.2.4 Learning Policies and Performance Measure

An online active learning strategy  $\pi$  consists of a sequence of query rules  $\{\nu_t\}_{t \geq 1}$  and a sequence of prediction rules  $\{\lambda_t\}_{t \geq 1}$ . Here  $\nu_t$  and  $\lambda_t$  map from causally available information consisting of past actions, instances, and queried labels to, respectively, the query decision of 0 (no query) or 1 (query) and a predicted label at time  $t$ . With a slight abuse of notation, we also let  $\nu_t$  and  $\lambda_t$  denote the resulting query decision and the predicted label at time  $t$  under these respective rules.

The performance of policy  $\pi = (\{\nu_t\}, \{\lambda_t\})$  over a horizon of length  $T$  is measured by the expected number of queries and the expected number of classification errors in excess to that of the Bayes optimal classifier  $h^*$ . These two performance measures, referred to as label complexity  $\mathbb{E}[Q(T)]$  and regret  $\mathbb{E}[R(T)]$ , are given as follows.

$$\mathbb{E}[Q(T)] = \mathbb{E} \left[ \sum_{t=1}^T \mathbb{1}[\nu_t = 1] \right] \quad (7.8)$$

$$\mathbb{E}[R(T)] = \mathbb{E} \left[ \sum_{t \leq T: \nu_t = 0} \mathbb{1}[\lambda_t \neq Y_t] - \mathbb{1}[h^*(X_t) \neq Y_t] \right], \quad (7.9)$$

where the expectation is with respect to the stochastic process induced by  $\pi$ . Note that regret measures the expected difference in the *cumulative* classification errors over the entire horizon between a learner employing  $\pi$  and an oracle that uses  $h^*$  all through the horizon.

The objective is a learning algorithm that minimizes the label complexity  $\mathbb{E}[Q(T)]$  with a constraint on the regret  $\mathbb{E}[R(T)]$ . The constraint, for example, can be either bounded by a constant independent of  $T$  or in a logarithmic order of  $T$ .

## 7.3 The Online Active Learning Algorithm

### 7.3.1 The Basic Structure

The algorithm operates under an epoch structure. When a fixed number  $M$  of labels have been queried in the current epoch, this epoch ends and the next one starts. Note that the epoch length, lower bounded by  $M$ , is random due to the real-time active query decisions. The algorithm maintains two sets in each epoch  $k$ : the version space  $\mathcal{H}_k$  and the RoD  $\mathcal{D}(\mathcal{H}_k)$  defined as the region of instances for which there is disagreement among hypotheses in the current version space  $\mathcal{H}_k$ . More specifically,

$$\mathcal{D}(\mathcal{H}_k) = \{x \in \mathcal{X} : \exists h_1, h_2 \in \mathcal{H}_k, h_1(x) \neq h_2(x)\}. \quad (7.10)$$

The initial version space is set to the entire hypothesis space  $\mathcal{H}$ , and the initial RoD is the instance space  $\mathcal{X}$ . At the end of each epoch, these two sets are updated using the  $M$  labels obtained in this epoch, and the algorithm iterates into the next epoch.

At each time instant  $t$  of epoch  $k$ , the query and prediction decisions are as follows. If  $x_t \in \mathcal{D}(\mathcal{H}_k)$ , its label is queried. Otherwise, the learner predicts the label of  $x_t$  using an arbitrary hypothesis in  $\mathcal{H}_k$ .

At the end of the epoch,  $\mathcal{H}_k$  is updated as follows. Let  $\mathcal{Z}_k$  denote the set of the  $M$  queried examples in this epoch. For a hypothesis  $h$  in  $\mathcal{H}_k$ , define its empirical error over  $\mathcal{Z}_k$  as

$$\epsilon_{\mathcal{Z}_k}(h) = \frac{1}{M} \sum_{(x,y) \in \mathcal{Z}_k} \mathbb{1}[h(x) \neq y]. \quad (7.11)$$

Let  $h_k^* = \arg \min_{h \in \mathcal{H}_k} \epsilon_{\mathcal{Z}_k}(h)$  be the best hypothesis in  $\mathcal{H}_k$  in terms of empirical error over  $\mathcal{Z}_k$ . The version space is then updated by eliminating each hypothesis  $h$  whose empirical error over  $\mathcal{Z}_k$  exceeds that of  $h_k^*$  by a threshold  $\Delta_{\mathcal{Z}_k}(h, h_k^*)$  that is specific to  $h$ ,  $h_k^*$ , and  $\mathcal{Z}_k$ . Specifically,

$$\mathcal{H}_{k+1} = \{h \in \mathcal{H}_k : \epsilon_{\mathcal{Z}_k}(h) - \epsilon_{\mathcal{Z}_k}(h_k^*) < \Delta_{\mathcal{Z}_k}(h, h_k^*)\}. \quad (7.12)$$

The new RoD  $\mathcal{D}(\mathcal{H}_{k+1})$  is then determined by  $\mathcal{H}_{k+1}$  as in (7.10).

### 7.3.2 Threshold Design

We now discuss the key issue of designing the threshold  $\Delta_{\mathcal{Z}_k}(h, h_k^*)$  for eliminating suboptimal hypotheses. This elimination threshold controls the tradeoff between two conflicting objectives: quickly shrinking the RoD (thus reducing label complexity) and managing the irreversible risk of eliminating good classifiers (thus increasing future classification errors).

In OLA, we obtain a more aggressive threshold design focusing on empirical errors incurred over significant  $(X, Y)$  examples determined by the current RoD. Specifically, for a pair of hypotheses  $h_1, h_2$ , define

$$\epsilon_{\mathbb{P}}(h_1, h_2) = \mathbb{P}(h_1(X) \neq Y \wedge h_2(X) = Y), \quad (7.13)$$

where  $\wedge$  is the logical AND operation. Thus,  $\epsilon_{\mathbb{P}}(h_1, h_2)$  is the probability that  $h_1$  misclassifies a random instance but  $h_2$  successfully classified. For a finite set  $\mathcal{Z}$

of  $(x, y)$  samples, the empirical excess error of  $h_1$  over  $h_2$  on  $\mathcal{Z}$  is defined as

$$\epsilon_{\mathcal{Z}}(h_1, h_2) \triangleq \frac{1}{|\mathcal{Z}|} \sum_{(x,y) \in \mathcal{Z}} \mathbb{1}[h_1(x) \neq y \wedge h_2(x) = y]. \quad (7.14)$$

The elimination threshold  $\Delta_{\mathcal{Z}_k}(h, h_k^*)$  is set to:

$$\begin{aligned} \Delta_{\mathcal{Z}_k}(h, h_k^*) &= \beta_{\mathcal{H}_k, M}^2 + \\ &\beta_{\mathcal{H}_k, M} \left( \sqrt{\epsilon_{\mathcal{Z}_k}(h, h_k^*)} + \sqrt{\epsilon_{\mathcal{Z}_k}(h_k^*, h)} \right), \end{aligned} \quad (7.15)$$

where  $\beta_{\mathcal{H}', n} = \sqrt{(4/n) \ln(16T^2 S(\mathcal{H}', 2n)^2)}$  for an arbitrary hypothesis space  $\mathcal{H}'$  and positive integer  $n$ . Here  $S(\mathcal{H}', n)$  is the  $n$ -th shattering coefficient of  $\mathcal{H}'$ . By Sauer's lemma [41],  $S(\mathcal{H}', n) = O(n^{d'})$  with  $d'$  being the VC dimension of  $\mathcal{H}'$ .

The choice of this specific threshold function will become clear in Section 7.4.1 when the relationship between the empirical error difference of two hypotheses and the ensemble error rate difference under  $\mathbb{P}$  is analyzed.

---

**Algorithm 17** The OLA Algorithm

---

**Input:** Time horizon  $T$ , VC dimension  $d$ , parameter  $m \in \mathbb{N}^+$ .  
**Initialization:** Set  $\mathcal{Z}_1 = \emptyset$ , Version space  $\mathcal{H}_1 = \mathcal{H}$ , RoD  $\mathcal{D}_1 = \mathcal{X}$ . Current epoch  $k = 1$ .  $M = \lceil mdT^{\frac{2-2\alpha}{2-\alpha}} \log T \rceil$ .  
**for**  $t = 1$  **to**  $T$  **do**  
  **if**  $x_t \notin \mathcal{D}_k$  **then**  
    Choose any  $h \in \mathcal{H}_k$  and label  $x_t$  with  $h(x_t)$ ;  
  **end if**  
  **if**  $x_t \in \mathcal{D}_k$  **then**  
    Query label  $y_t$  and let  $\mathcal{Z}_k = \mathcal{Z}_k \cup \{(x_t, y_t)\}$ ;  
    **if**  $|\mathcal{Z}_k| = M$  **then**  
      Update  $\mathcal{H}_{k+1}$  and  $\mathcal{D}_{k+1}$  according to Eqn.(7.10) and Eqn.(7.12) with the elimination threshold  $\Delta_{\mathcal{Z}_k}$  given in Eqn. (7.15);  
      Let  $k = k + 1$ ;  
    **end if**  
  **end if**  
**end if**  
**end for**

---

A detailed description of the algorithm is given in Algorithm 17. The algorithm parameter  $M$  is set to  $\lceil mdT^{\frac{2-2\alpha}{2-\alpha}} \log T \rceil$ , where  $m$  is a positive integer whose

value will be discussed in Section 7.4.2. We point out that while the horizon length  $T$  is used as an input parameter to the algorithm, the standard doubling trick can be applied when  $T$  is unknown.

## 7.4 Analysis of Regret and Label Complexity

We first develop the following concentration inequality in Theorem 7.4.1 to establish the relationship between the empirical error and ensemble error rate of any pair of hypotheses. The proof employs the normalized uniform convergence VC bound [309]. Details can be found in Appendix F.1.

**Theorem 7.4.1.** *Let  $\mathcal{Z}$  be a set of  $n$  i.i.d.  $(X, Y)$ -samples under distribution  $\mathbb{P}$ . For all  $h_1, h_2 \in \mathcal{H}$ , we have, with probability at least  $1 - \delta$ ,*

$$\begin{aligned} \epsilon_{\mathbb{P}}(h_1) - \epsilon_{\mathbb{P}}(h_2) &\leq \epsilon_{\mathcal{Z}}(h_1) - \epsilon_{\mathcal{Z}}(h_2) \\ &+ \gamma_n^2 + \gamma_n(\sqrt{\epsilon_{\mathcal{Z}}(h_1, h_2)} + \sqrt{\epsilon_{\mathcal{Z}}(h_2, h_1)}), \end{aligned} \tag{7.16}$$

where  $\gamma_n = \sqrt{(4/n) \ln(8S(\mathcal{H}, 2n)^2/\delta)}$ .

Since all samples in  $\mathcal{D}_k$  are queried at epoch  $k$  in the proposed OLA algorithm, we can see that  $\mathcal{Z}_k$  is an i.i.d. sample of size  $M$  from distribution  $\mathbb{P}|\mathcal{D}_k$ , which is defined as

$$\mathbb{P}|\mathcal{D}_k(x) = \begin{cases} \mathbb{P}(x)/\phi(\mathcal{D}_k) & \text{if } x \in \mathcal{D}_k \\ 0 & \text{otherwise} \end{cases}, \tag{7.17}$$

where  $\phi(\mathcal{D}) = \mathbb{P}(X \in \mathcal{D})$  for  $\mathcal{D} \subseteq \mathcal{X}$ .

Therefore, we can apply Theorem 7.4.1 to each epoch  $k$  with  $\mathcal{Z}_k$  and  $\mathbb{P}|\mathcal{D}_k$ , which gives us the following corollary.

**Corollary 7.4.2.** Let  $\beta_n = \sqrt{(4/n) \ln(16T^2 \mathcal{S}(\mathcal{H}, 2n)^2)}$ . With probability at least  $1 - \frac{1}{2T}$ , for all  $k \geq 1$  and for all  $h \in \mathcal{H}_k$ , we have

$$\begin{aligned} \epsilon_{\mathbb{P}|\mathcal{D}_k}(h_k^*) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h) &\leq \epsilon_{\mathcal{Z}_k}(h_k^*) - \epsilon_{\mathcal{Z}_k}(h) + \beta_M^2 \\ &\quad + \beta_M \left( \sqrt{\epsilon_{\mathcal{Z}_k}(h_k^*, h)} + \sqrt{\epsilon_{\mathcal{Z}_k}(h, h_k^*)} \right). \end{aligned} \tag{7.18}$$

### 7.4.1 Regret

Next, using Theorem 7.4.1 we show that the expected regret of the proposed OLA algorithm is bounded by  $1/2$ .

**Theorem 7.4.3.** The expected regret  $\mathbb{E}[R(T)]$  of the OLA algorithm is bounded as follows:

$$\mathbb{E}[R(T)] \leq \frac{1}{2}.$$

*Proof.* Here we provide the sketch of the proof. The detailed proof can be found in Appendix F.2 First we show that whenever we have  $h^* \in \mathcal{H}_k$ , the regret incurred by OLA in that epoch is 0. Using Corollary 7.4.2 we can conclude that  $h^* \in \mathcal{H}_k$  holds for all  $k \geq 1$  with probability at least  $1 - 1/(2T)$ , implying  $\mathbb{P}(R(T) > 0) \leq \frac{1}{2T}$ . By noting that  $R(T) \leq T$ , we have  $\mathbb{E}[R(T)] \leq \frac{1}{2T} \cdot T = \frac{1}{2}$  as desired.  $\square$

### 7.4.2 Label Complexity

For the purpose of label complexity analysis, we define the following online disagreement coefficient, which is slightly different from the disagreement coefficient defined for offline active learning in [125]. Recall the psuedo-metric  $\rho$

defined in (7.2). The online disagreement coefficient  $\theta = \theta(\mathbb{P}, \mathcal{H})$  is defined as

$$\theta = \sup \left\{ \frac{\phi[\mathcal{D}(B(h^*, r))]}{r} : r > 0 \right\}, \quad (7.19)$$

where  $B(h, r) = \{h \in \mathcal{H} : \rho(h, h') < r\}$  is a “hypothesis ball” centered at  $h$  with radius  $r$ .

The quantity  $\theta$  bounds the rate at which the disagreement mass of the ball  $B(h^*, r)$  grows with the radius  $r$ . It is bounded by  $\sqrt{d}$  when  $\mathcal{H}$  is  $d$ -dimensional homogeneous separators [125].

Next we upper bound the label complexity for the proposed online active learning algorithm.

**Theorem 7.4.4.** *Let  $\mathbb{E}[Q(T)]$  be the expected label complexity of OLA. If  $m > 324(\theta c_0)^{\frac{2}{\alpha}}$ , then there exists  $C_1 > 0$  such that*

$$\mathbb{E}[Q(T)] \leq C_1 m d T^{\frac{2-2\alpha}{2-\alpha}} (\log T + 1)^2, \quad (7.20)$$

where  $\theta = \theta(\mathbb{P}_X, \mathcal{H})$  is the disagreement coefficient.

Note that  $m$  is a constant determined by the algorithm, the label complexity  $\mathbb{E}[Q(T)]$  has an order of  $O(d T^{\frac{2-2\alpha}{2-\alpha}} \log^2 T)$ . For the Massart noise condition at  $\alpha = 1$ , the label complexity is  $O(d \log^2 T)$ .

We have discussed in Section 7.1.2 the key ideas and techniques used in the proof. The detailed proof can be found in Appendix F.3.

### 7.4.3 Order Optimality

We now establish the order optimality of the label complexity (upto polylogarithmic factors) of OLA under a bounded regret constraint. This is obtained

by establishing a lower bound on the label complexity feasible under any policy with a bounded regret.

**Theorem 7.4.5.** *Consider the Tsybakov noise satisfying the following condition with a parameter  $\alpha \in (0, 1)$ : there exist constants  $c_1$  and  $c_2$  independent of  $x \in \mathcal{X}$  such that  $\frac{c_1}{2}r_0(x)^{\frac{1}{\alpha}-1} \leq |\eta(x) - \frac{1}{2}| \leq \frac{c_2}{2}r_0(x)^{\frac{1}{\alpha}-1}$  holds for all  $x \in \mathcal{X}$  where  $r_0(x) = \inf_{\{h: x \in \mathcal{D}(h, h^*)\}} \rho(h, h^*)$ . The label complexity of all policies with bounded regret is of order  $\Omega(T^{\frac{2-2\alpha}{2-\alpha}})$ .*

Note that a lower bound on the noise (i.e., an upper bound specified through the constant  $c_2$  on the slope of  $\eta(x)$  passing  $1/2$ ) is further imposed in order to establish a tight lower bound on label complexity for a specific noise level. We point out that while we focused on the constraint of a bounded regret, the analysis can be easily modified to obtain lower bounds under regret constraints of different orders. Specifically, we can show a lower bound of  $\Omega(\min\{T^{2(1-\alpha)(1-\epsilon)}, T^{\frac{2-2\alpha}{2-\alpha}}\})$  under a regret constraint of order  $O(T^\epsilon)$  for some  $\epsilon > 0$ . It is also straightforward to modify the lower bound analysis to accommodate different problem models (e.g., those studied in [59, 333]).

*Proof.* The key in establishing the lower bound is to identify a limiting subproblem inherent to the online classification problem that determines the label complexity. We show that an inherent binary hypothesis testing problem presents such a limit. For this specific subproblem, we show that the probability of the event where label complexity is capped at  $\Omega(T^{\frac{2-2\alpha}{2-\alpha}})$  is small if the regret on the subproblem has to be bounded. The detailed proof is given in Appendix F.4.  $\square$

For the case of Massart Noise, we can establish a lower bound of  $\Omega(\log T)$  under the constraint of a sublinear regret budget. The basic proof technique

follows similar ideas as that for the Tsybakov noise but with a simplified analysis (See Appendix F.4). We summarize the lower bound for the case of Massart Noise in the following theorem.

**Theorem 7.4.6.** *Consider the Massart Noise model where  $\eta(x)$  is bounded away from  $1/2$  by a parameter  $\gamma > 0$ , i.e., for all  $x \in \mathcal{X}$ ,  $|\eta(x) - 1/2| \geq \gamma$ . The label complexity of all policies that achieve a sublinear regret under the above Massart Noise model is of the order  $\Omega(\log T)$ .*

For constraints of sublinear but unbounded regret, the above lower bound is tight since it matches with the upper bound on the label complexity of Random Walk OLA or RW-OLA for short (see Section 7.5). Under the constraint of bounded regret, however, we conjecture a  $\Omega(\log^2 T)$  lower bound on label complexity for a general hypothesis space with an infinite number of hypotheses. The intuition behind this conjecture is as follows. Let  $N(\epsilon)$  denote the  $\epsilon$ -covering number of  $\mathcal{H}$  and  $C$  be an associated  $\epsilon$ -cover that has  $N(\epsilon)$  hypotheses. Specifically, an  $\epsilon$ -cover  $C$  of  $\mathcal{H}$  is a subset of hypothesis  $\{h_1, \dots, h_N\}$  such that for any  $h \in \mathcal{H}$  there exists an  $i \in \{1, 2, \dots, N\}$  such that  $\rho(h, h_i) \leq \epsilon$  and the  $\epsilon$ -covering number of  $\mathcal{H}$  is the size of the smallest  $\epsilon$ -cover of  $\mathcal{H}$ . Following the same line of arguments in the proof of Theorem 7.4.6, we can show that for a hypothesis  $h \in C$ , the policy needs to query  $\Omega(\log T)$  instances in  $\mathcal{D}(h, h^*)$  to ensure a bounded regret and this needs to hold simultaneously for all  $h \in C$  by the end of the time horizon. Using the uniformity of the cover  $C$ , we can show that the expected number of queries to hit  $\Omega(\log T)$  queries in  $\mathcal{D}(h, h^*)$  for all  $h \in C$  is  $\Omega(\log T \log N(\epsilon))$ . Choosing  $\epsilon \sim T^{-1/2}$  results in a bound of  $\Omega(\log^2 T)$  on label complexity.

## 7.5 Extensions and Discussions

### 7.5.1 Tradeoff Between Label Complexity and Regret

In this section, we show that OLA can be modified to operate on a different point on the tradeoff curve of regret vs. label complexity.

In OLA, the threshold for elimination is constructed conservatively to achieve a bounded regret. More specifically, the outage probability of eliminating  $h^*$  from the version space (i.e., the parameter  $\delta$  in Theorem 7.4.1) in each epoch is capped at a small value  $1/T^2$  that diminishes with  $T$ . We now consider a variant of OLA in which the elimination probability  $\delta$  is set to a constant in order to quickly shrink RoD for a lower label complexity. To mitigate the high probability of  $h^*$  being eliminated, which may result in a linear regret order, we build in a verification stage at the beginning of each epoch for the algorithm to self recognize and recover from the event of  $h^*$  being eliminated. The key idea is to devise a biased random walk on the version spaces that allows the algorithm to trace back to a previous version space in the event of  $h^*$  being eliminated. The bias of the random walk, however, ensures that with high probability the trajectories of the random walk concentrate on subsets of  $\mathcal{H}$  containing  $h^*$  with disagreement regions diminishing at a desired rate. As a result, after taking  $O(\log T)$  steps, the random walk arrives at a version space with a disagreement region in the order of  $O(1/T)$  in terms of its probability mass. In summary, compared with a deterministic transition between consecutive version spaces as in OLA, the random walk approach allows us to move more swiftly on average across version spaces with the option of tracing back and hence correcting previous erroneous moves. It is crucial to the objective of operating at a different

point on the label complexity-regret curve. We refer to this variant of OLA as RW-OLA.

Before delving into the details of the verification stage, we define the parent and child relationship between version spaces. For each epoch  $k$ , if  $\mathcal{H}_k$  is obtained by eliminating some of the hypotheses in the version space  $\mathcal{H}_{r(k)}$  of a previous epoch  $r(k)$ , we say that  $\mathcal{H}_{r(k)}$  is the parent of  $\mathcal{H}_k$  and  $\mathcal{H}_k$  is a child of  $\mathcal{H}_{r(k)}$ . Note that  $\mathcal{H}_k$  is a subset of  $\mathcal{H}_{r(k)}$ .

**Verification Stage** In the verification stage of epoch  $k$ , the query and prediction decision are based on its parent version space  $\mathcal{H}_{r(k)}$  and its corresponding RoD. When a fixed number  $M$  of labels have been queried, we start the verification process as follows.

Let  $\mathcal{Z}'_k$  denote the set of the  $M$  queried examples in the verification stage. We examine two values in terms of the empirical error over  $\mathcal{Z}'_k$ : (1)  $\min_{h \in \mathcal{H}_k} \epsilon_{\mathcal{Z}'_k}(h)$ : the minimum empirical error inside  $\mathcal{H}_k$ ; (2)  $\min_{h \notin \mathcal{H}_k} \epsilon_{\mathcal{Z}'_k}(h)$ : the minimum empirical error outside  $\mathcal{H}_k$ . Intuitively, if  $h^* \in \mathcal{H}_k$ , the difference

$$\min_{h \notin \mathcal{H}_k} \epsilon_{\mathcal{Z}'_k}(h) - \min_{h \in \mathcal{H}_k} \epsilon_{\mathcal{Z}'_k}(h) \quad (7.21)$$

will be large, and vice versa. We hence determine the outcome of the verification stage based on whether this gap between the empirical errors outside and inside the current version space  $\mathcal{H}_k$  exceeds a properly designed threshold. If the verification passes, indicating that  $h^* \in \mathcal{H}_k$  with a sufficiently high probability, the epoch proceeds in the same way as in OLA, and the current version space  $\mathcal{H}_k$  is further pruned to form a new version space  $\mathcal{H}_{k+1}$ . If, on the other hand, the verification fails, the current epoch ends, and the algorithm traces back to the parent of  $\mathcal{H}_k$  by setting  $\mathcal{H}_{k+1} = \mathcal{H}_{r(k)}$ . The evolution of the version spaces across

epochs follows a biased random walk as detailed below.

**Random Walk on a Version-Space Tree** Based on the outcome of the verification stage, the version space  $\mathcal{H}_{k+1}$  of epoch  $k + 1$  is either a child or a parent of  $\mathcal{H}_k$ . In particular, following the evolution of the version spaces, we can construct a growing tree to record the parent-children relationship between version spaces. In this tree structure, each node represents a version space, and the version space sequence  $\{\mathcal{H}_k\}_{k \geq 1}$  forms a random walk on the tree. Illustrated in Fig. 7.1 is a sample path of the random walk on a tree for a hypothesis space consisting of threshold classifiers  $\mathcal{H} = \{h_z | 0 \leq z \leq 1\}$  on  $\mathcal{X} = [0, 1]$  where  $h_z = [z, 1]$ . We can see that on this tree, the verification failed in epochs 2 and 5 and but passed in epochs 1, 3, 4, and 6.

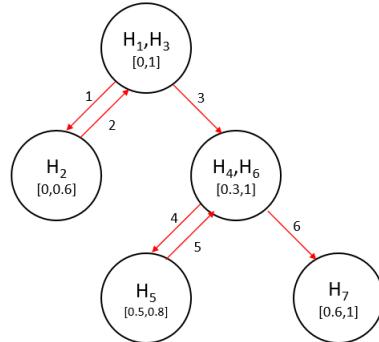


Figure 7.1: A typical random walk on a version space tree.

**Threshold design** In addition to the elimination threshold for pruning the version space as in OLA, RW-OLA also requires a verification threshold. As explained below, these two thresholds are coupled and need to be designed jointly to ensure the desired performance of the algorithm.

The verification stage performs a binary detection problem: whether  $h^*$  is

inside the current version space  $\mathcal{H}_k$ . On one hand, to ensure that the random walk on the version spaces is biased toward the direction of correct pruning, the verification threshold needs to be chosen to ensure a sufficiently accurate detection outcome. On the other hand, the hardness of this detection problem is determined by how close the two cases of  $h^* \in \mathcal{H}_k$  and  $h^* \notin \mathcal{H}_k$  are. More specifically, the hardness of this binary detection problem is determined by the error rate difference between the best hypothesis inside  $\mathcal{H}_k$  and the best hypothesis outside  $\mathcal{H}_k$ :

$$\min_{h \notin \mathcal{H}_k} \epsilon_{\mathbb{P}_{r(k)}}(h) - \min_{h \in \mathcal{H}_k} \epsilon_{\mathbb{P}_{r(k)}}(h), \quad (7.22)$$

which, in turn, is determined by the elimination threshold in epoch  $r(k)$  when  $\mathcal{H}_k$  is obtained. More specifically, while a smaller elimination threshold leads to more aggressive pruning of the version space, it results in a smaller performance gap between hypotheses outside  $\mathcal{H}_k$  and those inside  $\mathcal{H}_k$  since near-optimal hypotheses may be eliminated from the version space. Consequently, the verification stage faces a harder detection problem. In summary, the two thresholds need to be designed jointly to balance the label complexity associated with verification and with normal learning.

Let  $p$  denote the desired bias of the random walk. This implies that (i) when  $h^* \in \mathcal{H}_k$ , the verification passes with a probability no smaller than  $p$ ; (ii) when  $h^* \notin \mathcal{H}_k$ , the verification fails with a probability no smaller than  $p$ . Let

$$\Delta(M, \delta) = 2 \sqrt{2 \frac{\log \mathcal{S}(\mathcal{H}, 2M) + \log \frac{2}{\delta}}{M}}. \quad (7.23)$$

We set the elimination threshold to  $6\Delta(M, 1 - \sqrt{p})$  so that the error rate difference in Eqn. 7.22, which determines the hardness of the verification problem,

is at least  $4\Delta(M, 1 - \sqrt{p})$  with high probability. The verification threshold is set to  $2\Delta(M, 1 - \sqrt{p})$  to guarantee the desired bias of  $p$ . A detailed derivation of the thresholds is given in Appendix F.5.

---

**Algorithm 18** The Random Walk OLA (RW-OLA) Algorithm

---

**Input:** VC dimension  $d$ , parameter  $m \in \mathbb{N}^+$ .  
**Initialization:** Set Version space  $\mathcal{H}_1 = \mathcal{H}$ , RoD  $\mathcal{D}_1 = \mathcal{X}$ . Current epoch  $k = 1$ .  $M = md$ . Parents  $r(1) = 1$

**while**  $t \leq T$  **do**

**Verification:**

Let  $Z'_k = \emptyset$

**while**  $|Z'_k| < M$  **do**

Let  $t = t + 1$

**if**  $x_t \notin \mathcal{D}_{r(k)}$  **then**

Choose any  $h \in \mathcal{H}_{r(k)}$  and label  $x_t$  with  $h(x_t)$ ;

**else**

Query label  $y_t$  and let  $\mathcal{Z}'_k = \mathcal{Z}'_k \cup \{(x_t, y_t)\}$ ;

**end if**

**end while**

**if**  $\min_{h \notin \mathcal{H}_k} \epsilon_{\mathcal{Z}'_k}(h) - \min_{h \in \mathcal{H}_k} \epsilon_{\mathcal{Z}'_k}(h) < 2\Delta(M, 1 - p)$  **then**

Let  $\mathcal{H}_{k+1} = \mathcal{H}_{r(k)}$ ,  $\mathcal{D}_{k+1} = \mathcal{D}_{r(k)}$ ,  $r(k+1) = r(r(k))$ ,  $k \leftarrow k + 1$ ;  
 continue;

**end if**

**Elimination:**

Let  $Z_k = \emptyset$

**while**  $|Z_k| < M$  **do**

Let  $t = t + 1$

**if**  $x_t \notin \mathcal{D}_k$  **then**

Choose any  $h \in \mathcal{H}_k$  and label  $x_t$  with  $h(x_t)$ ;

**else**

Query label  $y_t$  and let  $\mathcal{Z}_k = \mathcal{Z}_k \cup \{(x_t, y_t)\}$ ;

**end if**

**end while**

Update  $\mathcal{H}_{k+1}$  as following:

$$\mathcal{H}_{k+1} = \{h \in \mathcal{H}_k : \epsilon_{\mathcal{Z}_k}(h) - \epsilon_{\mathcal{Z}_k}(h_k^*) < 6\Delta(M, 1 - \sqrt{p})\} \quad (7.24)$$

Update  $\mathcal{D}_{k+1}$  according to (10).  
 Let  $r(k+1) = k$ ,  $k = k + 1$ ;  
**end while**

---

A detailed description of the algorithm is given in Algorithm 18. The algorithm parameter  $M$  is set to  $\lceil md \rceil$ , where  $m$  is a positive integer whose value will be discussed in the analysis below.

### Analysis of Regret and Label Complexity

**Theorem 7.5.1.** *Let  $\mathbb{E}[Q(T)]$  be the expected label complexity of the RW-OLA algorithm. If  $m > 1024(\theta c_0)^2$ , under Massart noise condition, there exists  $C_2 > 0$  such that*

$$\mathbb{E}[Q(T)] \leq C_2 m d \log T, \quad (7.25)$$

where  $\theta = \theta(\mathbb{P}_X, \mathcal{H})$  is the disagreement coefficient.

*Proof.* Here we provide a sketch of the proof. The detailed proof can be found in Appendix F.5. We first show that the bias of the random walk is indeed bounded above  $p$  with the chosen thresholds. We then show that when the verification passes, the RoD in the next epoch will shrink with a fixed rate  $c$ . Based on these two statements, we can show that the expected RoD is decreasing exponentially with rate  $c_1 = 1 - p + c^2 p < 1$ . The same submartingale technique used in analyzing OLA is then used to bound the label complexity of RW-OLA.  $\square$

Under Massart noise, the epoch length for OLA is  $md \log T$ , which leads to  $O(\log^2 T)$  label complexity. For RW-OLA, the epoch length is only  $md$ , which makes the label complexity only  $O(\log T)$ . In other words, to make sure RoD decreases exponentially, OLA requires epoch length to be  $O(d \log T)$  but RW-OLA only requires it to be  $O(1)$ .

**Theorem 7.5.2.** *Let  $\mathbb{E}[R(T)]$  be the expected regret of the RW-OLA algorithm. If  $m >$*

$1024(\theta'c_0)^2$  and  $\theta' > 0$ , under Massart noise condition, there exists  $C_3 > 0$  such that

$$\mathbb{E}[R(T)] \leq C_3 m d \log T, \quad (7.26)$$

where

$$\theta' = \inf \left\{ \frac{\phi[\mathcal{D}(B(h^*, r))]}{r} : r > 0 \right\}. \quad (7.27)$$

is the modified disagreement coefficient.

*Proof.* Since an epoch  $k$  with  $h^* \in \mathcal{H}_k$  incurs no regret, we only need to consider the case where  $h^* \notin \mathcal{H}_k$ . In this case, based on the RW-OLA algorithm, regret incurs if and only if the instance falls into a subset outside of RoD. Based on the bias of the random walk, we can show that the expected ratio of that subset to the current RoD is bounded by a constant. Since queries occur whenever the instances fall inside the RoD, we can show that the expected regret is upper bounded by this constant multiplying the expected label complexity, which is  $O(d \log T)$ . Please refer to Appendix F.6 for the detailed proof.  $\square$

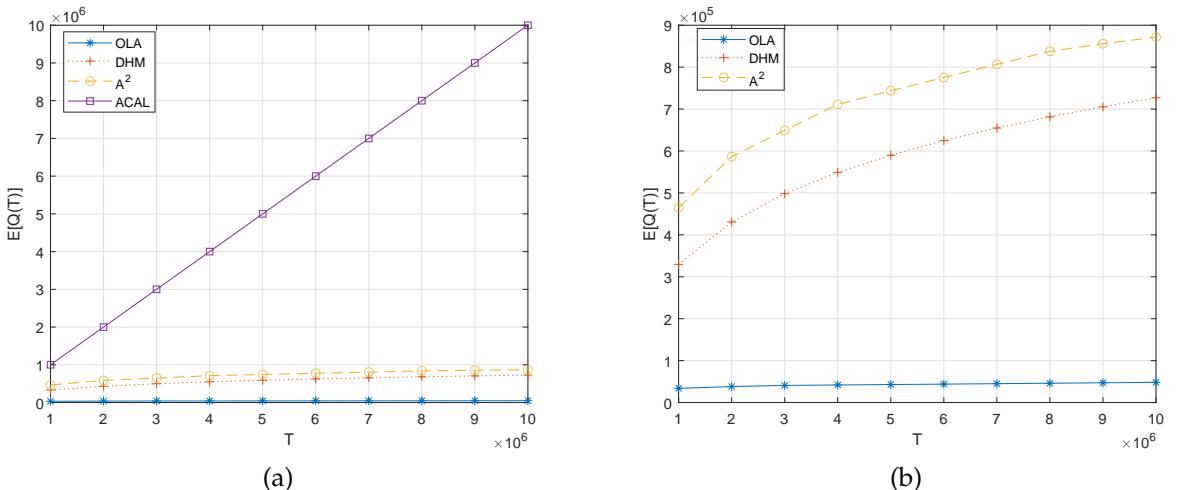


Figure 7.2: Comparison with  $A^2$ , DHM, and ACAL ( $d = 1$ , Tsybakov noise with  $\alpha = 1$  and  $c_0 = 5$ ,  $h^* = h_{0.5}$ ).

### 7.5.2 Implementation for Homogeneous Linear Classification

There are several steps in OLA and RW-OLA that can be computationally expensive, which is inherent to the disagreement-based approach. Specifically, maintaining the version space and RoD, and computing the best empirical hypothesis  $h_k^*$  can be costly. We discuss here approximate implementations with manageable computational complexity for homogeneous linear classification, drawing inspiration from techniques of using surrogate loss [129] and the Query-by-Committee approaches [111, 85].

In homogeneous linear classification,  $\mathcal{X}$  is the surface of the  $d$ -dimension unit Euclidean sphere. Each hypothesis, as a linear separator that passes the origin, is given by a unit vector  $\mathbf{u} \in \mathbb{R}^d$  such that the corresponding concept is  $\{\mathbf{x} \in \mathcal{X} : \mathbf{u}\mathbf{x} \geq 0\}$ .

To estimate the best empirical hypothesis  $h_k^*$ , we use hinge loss function  $l(z) = \max\{1 - z, 0\}$  to replace the 0-1 loss function in Eqn. (7.11). Then, the best empirical hypothesis  $\hat{h}_k^*$  under hinge loss is given by

$$\hat{h}_k^* = \min_{h \in \mathcal{H}_k} \sum_{(x,y) \in \mathcal{Z}_k} \max\{1 - (2y - 1)\mathbf{u}x, 0\}. \quad (7.28)$$

Then standard linear classification algorithms such as SVM can be employed to compute  $\hat{h}_k^*$ .

The version space is approximated with  $N$  constituent hypotheses sampled uniformly at random. Specifically, at  $t = 1$ , we sample  $N$  hypotheses uniformly at random from the entire hypothesis space  $\{\mathbf{u} \in \mathbb{R}^d, \|\mathbf{u}\| = 1\}$  and form  $\hat{\mathcal{H}}_1$ . At each epoch  $k$ , for each hypothesis  $h \in \hat{\mathcal{H}}_k$ , we check whether it should be eliminated based on Eqn. (7.12)) and label them as  $+1$  or  $-1$  accordingly. Then, we run a linear classification algorithm to separate the hypotheses labeled  $+1$

from those labeled  $-1$  in the above process. This yields a linear classifier of the form  $\omega\mathbf{u} + b \geq 0$  that represents an approximation of the new version space. To obtain an approximation of the new version space  $\hat{\mathcal{H}}_{k+1}$ , we again sample  $N$  hypotheses uniformly at random from  $\{\mathbf{u} : \omega\mathbf{u} + b \geq 0\}$  to form the next version space  $\hat{\mathcal{H}}_{k+1}$ .

Since the version space is estimated by a finite number of hypotheses, instead of maintaining the RoD explicitly, we check whether  $x_t \notin \mathcal{D}_k$  by checking whether all  $h \in \hat{\mathcal{H}}_k$  agree on  $x_t$ . A detailed description of the algorithm is given in Algorithm 19.

For RW-OLA, both the verification stage and elimination stage can be implemented similarly. In particular, the elimination stage of RW-OLA is exactly the same as OLA except the threshold will be different. Therefore, it can be done by replacing the threshold in step 2 to maintain the version space and RoD. For the verification stage, which involves finding the best empirical hypothesis inside and outside of  $\mathcal{H}_k$ , can be done using the hinge loss replacement in Eqn. (7.28) with a standard linear classification algorithm as well.

## 7.6 Simulation Examples

We first compare the label complexity of OLA and RW-OLA with existing disagreement-based active learning algorithms. We first consider a one-dimensional instance space  $X = [0, 1]$  and threshold classifiers with  $\mathcal{H} = \{h_z | 0 \leq z \leq 1\}$  where  $h_z = [z, 1]$ . Note that the VC dimension  $d = 1$ . We set  $\mathbb{P}_X$  to be the uniform distribution. Figure 7.2 and 7.3 show the comparison under different Tsybakov noise conditions.

---

**Algorithm 19** OLA for Homogeneous Linear Classification

---

**Initialization:** Set  $\mathcal{Z}_0 = \emptyset$ , Random sample  $\hat{\mathcal{H}}_0$  uniformly from  $\mathcal{H}$ .  
**for**  $t = 1$  to  $T$  **do**  
  **if** All  $h \in \hat{\mathcal{H}}_k$  agree on  $x_t$  **then**  
    Choose any  $h \in \hat{\mathcal{H}}_k$  and label  $x_t$  with  $h(x_t)$ ;  
  **else**  
    Query label  $y_t$  and let  $\mathcal{Z}_k = \mathcal{Z}_k \cup \{(x_t, y_t)\}$ ;  
    **if**  $|\mathcal{Z}_k| = M$  **then**  
      1. Find  $h_k^*$  using  $\mathcal{Z}_k$  and Eqn. (7.28);  
      2. For all  $h \in \hat{\mathcal{H}}_k$ , check whether  $h \in \mathcal{H}_{k+1}$  based on Eqn. (7.12) and label them accordingly;  
      3. Find linear classifier  $\omega\mathbf{u} + b \geq 0$  for  $\hat{\mathcal{H}}_k$  and its label;  
      4. Random sample  $\hat{\mathcal{H}}_{k+1}$  from  $\{\mathbf{u} : \omega\mathbf{u} + b \geq 0\}$ ;  
    **end if**  
  **end if**  
**end for**

---

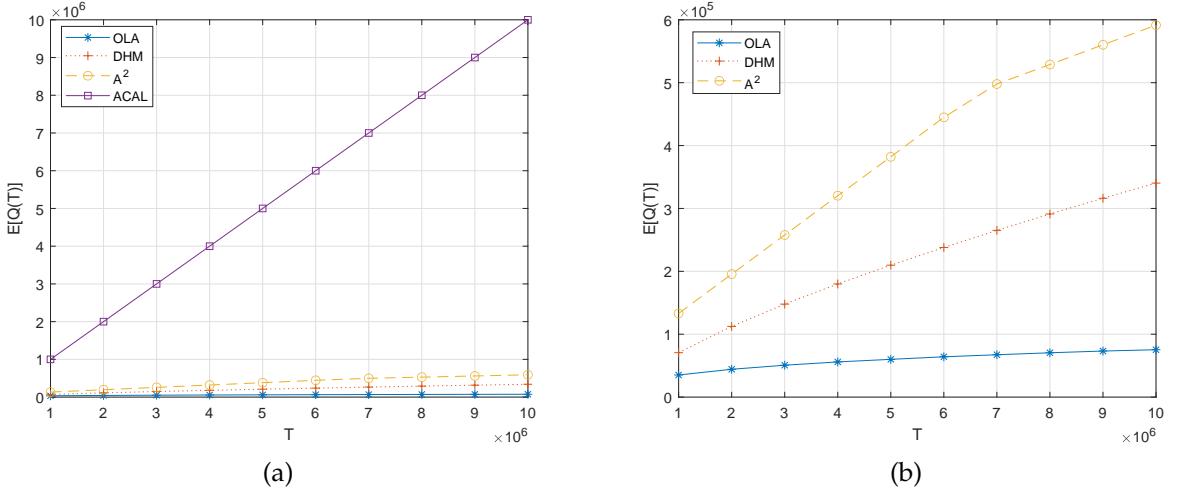


Figure 7.3: Comparison with  $A^2$ , DHM, and ACAL ( $d = 1$ , Tsybakov noise with  $\alpha = 0.5$  and  $c_0 = 1$ ,  $h^* = h_{0.5}$ ).

In Figure 7.4, we consider the same instance space  $X = [0, 1]$  and uniformly distributed instances, but a hypothesis space  $\mathcal{H} = \{h_{z_1, z_2} | 0 \leq z_1, z_2 \leq 1\}$  consisting of all intervals  $h_{z_1, z_2} = [z_1, z_2]$ . Note that in this case, the VC dimension  $d = 2$ . For both cases, since the hypothesis space can be effectively represented by  $N^d$  points (with  $d = 1, 2$ ) after taking  $N$  samples, we implemented an exact ver-

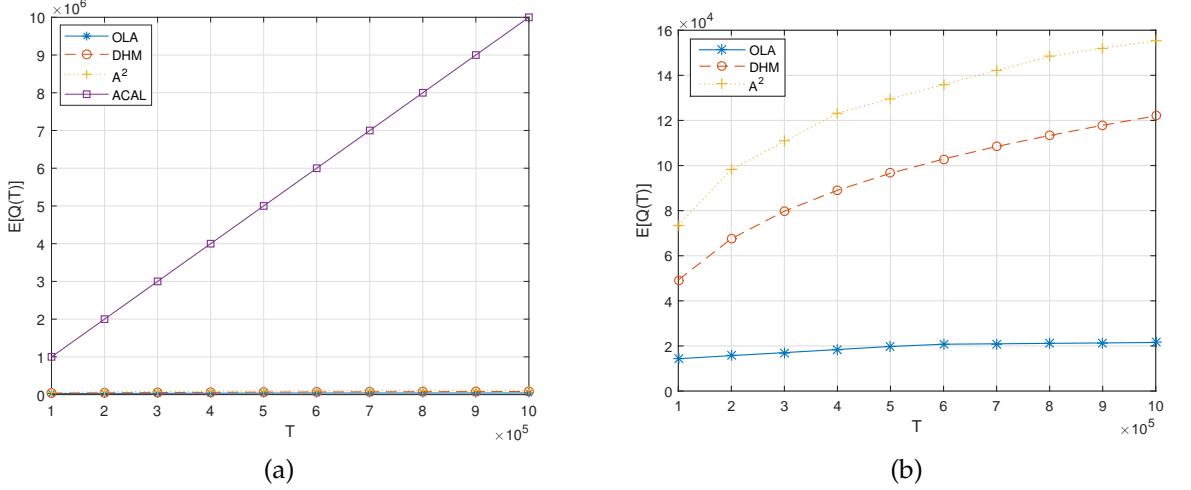


Figure 7.4: Comparison with  $A^2$ , DHM, and ACAL for ( $d = 2$ , Tsybakov  $\alpha = 1$  and  $c_0 = 1$   $h^* = h_{0,25,0.75}$ ).

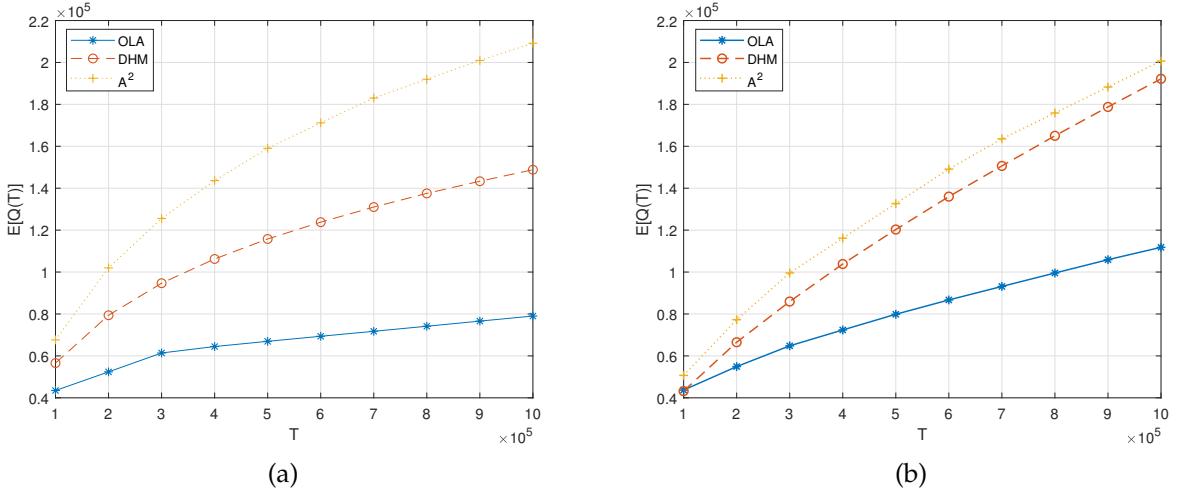


Figure 7.5: Comparison with  $A^2$  and DHM for ( $d = 3$ ,  $N = 50000$ , Tsybakov noise with  $\alpha = 1$  and  $c_0 = 1$ ,  $\alpha = 0.5$  and  $c_0 = 5$ ,  $h^* = (1, 0, 0)$ ).

sion of all the algorithms. For ACAL, the optimization problem to obtain the threshold was solved numerically.

Since the label complexity for ACAL is much larger than the others, we plot the others in the right figure. The significant reduction in label complexity offered by OLA and RW-OLA is evident from Figures 7.2-7.4. The simulated classification errors are near zero for all the algorithms.

Next we consider  $d$ -dimension homogeneous linear classification setting. Figure 7.5 and 7.6 show the comparison for  $d = 3, 4$ . DHM and  $A^2$  are implemented with similar methods as discussed in Section 7.5.2. The simulated classification errors are near zero for all three algorithms.

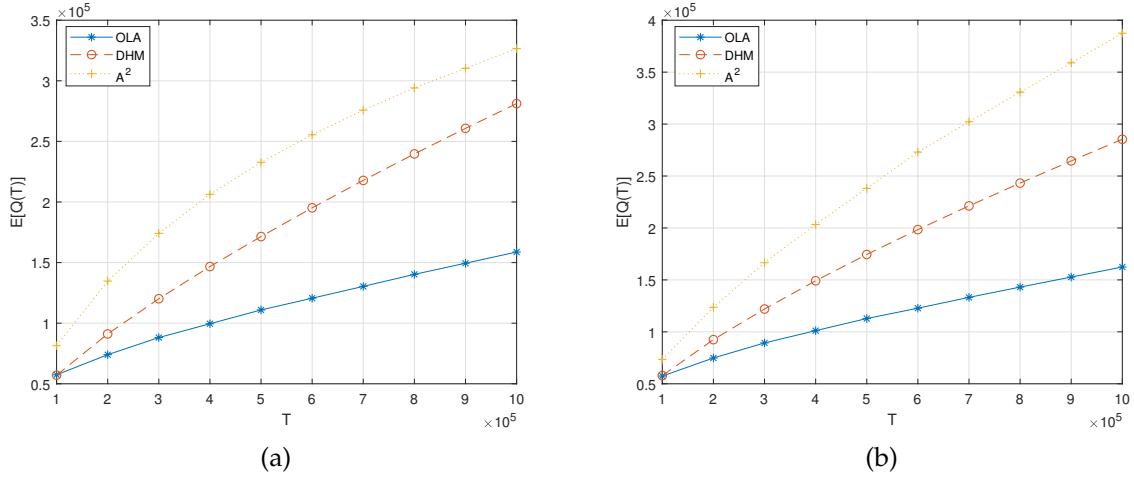


Figure 7.6: Comparison with  $A^2$  and DHM for ( $d = 4, N = 50000$ , Tsybakov noise with  $\alpha = 1$  and  $c_0 = 1$ ,  $\alpha = 0.5$  and  $c_0 = 5$   $h^* = (1, 0, 0, 0)$ ).

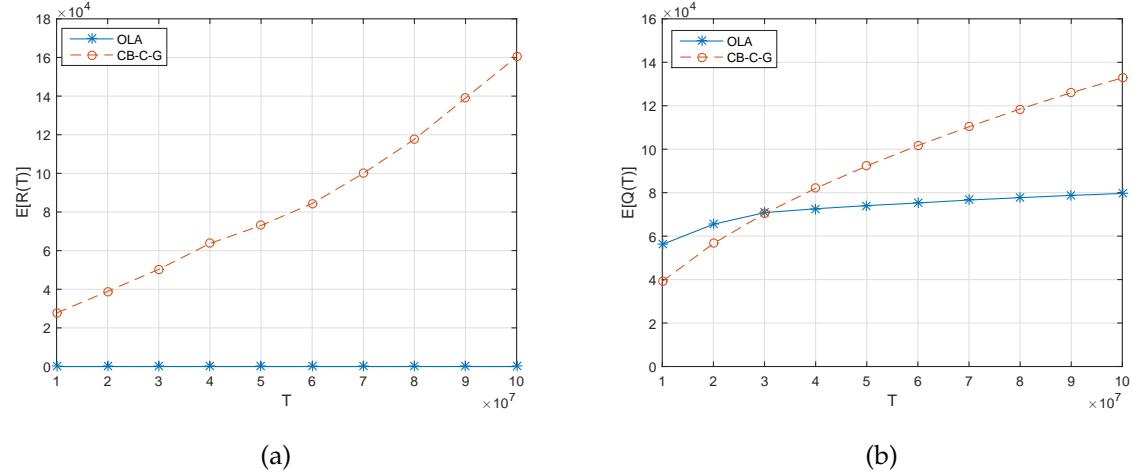


Figure 7.7: Comparison with CB-C-G [61] under Tsybakov noise with  $\alpha = 0.5$  and  $c_0 = 1$ .

Next we compare OLA with the online margin-based algorithm CB-C-G proposed in [61, 59]. It is specialized in learning homogeneous separators under specific noise model: there exists a fixed and unknown vector  $\mathbf{u} \in \mathbb{R}^d$  with

Euclidean norm  $\|\mathbf{u}\| = 1$  such that  $\eta(\mathbf{x}) = (1 + \mathbf{u}^\top \mathbf{x})/2$ . Then, the Bayes optimal classifier  $h^*(\mathbf{x}) = \mathbb{1}[\mathbf{u}^\top \mathbf{x} \geq 0]$ . Shown in Figure 7.7 are the label complexity and classification error comparisons under this specific noise model with  $d = 2$ ,  $\mathbf{u} = (1, 0)$ , and uniform  $\mathbb{P}_X$ . It shows that even when comparing under this special setting, OLA offers considerable reduction in label complexity and drastic improvement in classification accuracy. These simulation results suggest that the more conservative disagreement-based approach might be more suitable in the online setting than the aggressive margin-based approach, especially in terms of bounding the online classification errors. The reason is that the margin-based approaches aggressively seek out the most informative samples to query, which may have an advantage in the offline setting when unqueried samples can be skipped without labeling. In the online setting, however, such approaches result in more frequent self-labeling, hence higher rate of online classification errors.

# CHAPTER 8

## ADAPTIVE GROUP TESTING UNDER UNKNOWN NOISE MODELS

### 8.1 Introduction

Group testing is a common strategy employed to identify defective items from a given population of items. The idea of group testing, initiated by Robert Dorfman [98], involves performing tests over subsets of the population instead of tests on individual items to reduce the number of required tests. There is now a renewed interest in group testing to expedite testing for COVID-19 [104]. The objective under the group testing strategy is to outline a sequence of subsets to be tested so that all the defective items can be identified using a minimum possible number of tests.

A large fraction of existing studies on group testing assume error-free test outcomes. The works in the relatively slimmer body of literature of noisy group testing mainly focus on boolean group testing under special noise models like one-sided or symmetric noise and assume the knowledge of the statistics of the underlying noise model. With the increasing applications of group testing, the aforementioned assumptions need not hold in practice for various applications. Furthermore, obtaining an exact statistical model of the underlying noise is often expensive or infeasible. Thus, the objective in this chapter is to develop a group testing strategy that is applicable to a wider class of measurement models and is also agnostic to the exact noise statistics. This appears to be the first work considering noisy group testing under unknown noise models.

### 8.1.1 Main Results

We consider the problem of quantitative group testing (QGT) where the result of each test reveals the number of defective items in the tested group. The result of each test is assumed to be independently corrupted by noise with an unknown distribution.

We focus on adaptive testing strategies where tests are designed sequentially over time based on past test outcomes. We propose a new adaptive strategy, referred to as HRW-QGT, which is based on a hierarchy of biased random walks guided by a local confidence-based sequential test. We establish an order-optimal performance of HRW-QGT, both in terms of the size of the population and the error rate. The random walk structure lends adaptivity and optimal performance guarantees to the algorithm while the local sequential test lends the required agnosticism the unknown noise model.

The proposed confidence-based local sequential test might also be of independent interest for designing other group testing strategies.

#### Prior Work:

### 8.1.2 Prior Work

The concept of Quantitative Group Testing originated as the coin weighing problem introduced by Shapiro [266] and has been extensively studied over the years, especially for the case of  $d = 2$  [110, 105, 275, 97, 145, 9, 133, 113].

For the case of general  $d$ , Aigner and Schughart [10] obtained the performance guarantees of the optimal adaptive testing plan with a nested structure. Recently, Wang *et al.* [312] designed an adaptive nested strategy that achieves the optimal performance. Bshouty [43] outlined a series of adaptive test plans based on search and detection matrices, with polynomial running time. Karimi *et al.* [158] designed a non-adaptive scheme for QGT using sparse graph codes over bipartite graphs that was bettered by the greedy reconstruction algorithm in [114]. All these works assume error-free tests unlike the setting considered in this paper.

For QGT, the only work that consider noisy observations is by Bshouty and Mazzawi [44] where they assume that the results of certain tests can be erroneous/erased. For boolean group testing, several non-adaptive [62, 203, 182] and adaptive [47, 258, 259] algorithms have been proposed for noisy group testing. However, these algorithms rely on the knowledge of the underlying noise model. On the other hand, the proposed approach is applicable for a generic and an unknown noise model.

Lastly, the strategy proposed in this work is inspired by the idea of random walk on a tree first proposed in Wang *et al.* [311] for active hypothesis testing. It was later also extended for heavy hitter detection [307] and stochastic optimization [306]. In this work, we not only extend this idea for group testing but also the proposed strategy involves inducing a biased random walk on a more general graph as opposed to a tree. Furthermore, the local sequential test proposed in this work is different from the local tests considered in the above works.

## 8.2 Problem Formulation

We consider the problem of designing a group testing strategy,  $\pi$ , to identify the subset of  $d$  defective items,  $\mathcal{D}$ , from the population  $\mathcal{S} = \{1, 2, \dots, n\}$  of  $n$  items with a confidence of  $1 - \epsilon$ . A test on a set  $\mathcal{A} \subseteq \mathcal{S}$  reveals a noisy measurement of the number of defective items in  $\mathcal{A}$  given by  $|\mathcal{D} \cap \mathcal{A}|$ , where  $|\cdot|$  denotes the cardinality of a set. The noise model is designed to capture the incorrect estimation of the defective number of items using the testing procedure. In particular, let  $X = \{0, 1, 2, \dots, d\}$  denote the set of possible number of defectives in a test and let  $\mathcal{Y} = \{0, 1, 2, \dots, K\}$  denote the possible set of outcomes of a test, where  $K \geq d$ . The noise is modelled using a probability transition matrix  $P$ , assumed to be unknown, where  $P_{ij} = \Pr(Y = j|X = i)$  for  $i \in X$  and  $j \in \mathcal{Y}$  and  $X$  and  $Y$  denote the noiseless measurement and output of the test respectively. The result of each test on a subset  $\mathcal{A}$  with  $x$  defectives is assumed to be independent of the previous tests and is distributed as  $P(\cdot|x)$ .

We allow  $P$  to be any general transition matrix that satisfies the following mild assumption: for all  $x \in X$ ,  $\Pr(Y = x|X = x) > \Pr(Y = y|X = x) + \rho$  for all  $y \in \mathcal{Y} \setminus \{x\}$ . Here  $\rho > 0$  is a universal constant whose value is not assumed to be known. Such a noise model is fairly generic as it encompasses a large number of known models. For example, this model includes all the noisy boolean group test models where the probabilities of the false alarm and miss detection are bounded away from  $1/2$  and are even possibly different as opposed to the symmetric noise models generally considered in the literature.

We focus on the design of adaptive policies where tests are designed sequentially over time based on past test outcomes. Let  $\Pi_A$  denote the class of adaptive

policies for noisy QGT. Given a policy,  $\pi_A \in \Pi_A$ , its performance is measured using the worst case expected number of tests required by  $\pi_A$ , i.e.,

$$\bar{\tau}_{\pi_A}(n, \epsilon) = \max_{\mathcal{D}} \mathbb{E}[\tau_{\pi}(n, \epsilon, \mathcal{D})], \quad (8.1)$$

where  $\tau_{\pi_A}(n, \epsilon, \mathcal{D})$  is the random number of tests required by the policy  $\pi_A$  to correctly identify the subset of defective items  $\mathcal{D}$  from a population of  $n$  items with a confidence of at least  $1 - \epsilon$ . The expectation is taken only with respect to the noise. Let  $\bar{\tau}^*(n, \epsilon)$  denote the optimal adaptive policy, i.e.,  $\bar{\tau}^*(n, \epsilon) = \inf_{\pi_A \in \Pi_A} \bar{\tau}_{\pi_A}(n, \epsilon)$ . The objective in this work is to design an adaptive policy that offers an order-optimal performance both in terms of the population size,  $n$ , and the error rate,  $\epsilon$ . We say that a policy  $\pi_0 \in \Pi_A$  is order-optimal if the following holds for some universal constants  $C_1, C_2 > 0$ :

$$\lim_{n \rightarrow \infty} \frac{\bar{\tau}_{\pi_0}(n, \epsilon)}{\bar{\tau}^*(n, \epsilon)} \leq C_1 \quad \text{and} \quad \lim_{\epsilon \rightarrow 0^+} \frac{\bar{\tau}_{\pi_0}(n, \epsilon)}{\bar{\tau}^*(n, \epsilon)} \leq C_2. \quad (8.2)$$

### 8.3 Algorithm Description

In this section, we describe the HRW-QGT algorithm that consists of three components, the Outer Module, the Inner Module and the Local Sequential Test. The Outer Module involves inducing a random walk on a graph  $\mathcal{G}$ , whose vertices correspond to subsets of  $\mathcal{S}$ . This random walk is guided by the Inner Module so that its each step is biased towards a superset of  $\mathcal{D}$ . By identifying supersets of  $\mathcal{D}$  of decreasing cardinality, the random walk is designed to zoom in on  $\mathcal{D}$ . The Inner Module itself consists of another random walk which is guided by the Local Sequential Test. Given a set  $\mathcal{A} \subseteq \mathcal{S}$ , the Inner Module constructs a binary tree with  $d^2$  leaves corresponding to equal sized partitions of  $\mathcal{A}$ . Using several biased random walks on this tree, it identifies all such leaves that have at least

one defective item thereby obtaining a set  $\mathcal{A}' \subset \mathcal{A}$  that contains all the defective items of  $\mathcal{A}$ . This allows the Outer Module to move towards supersets  $\mathcal{D}$  with smaller cardinality every time. The working of the Outer and the Inner Module is illustrated in Figures 8.1 and 8.2. Lastly, the Local Sequential Test takes repeated measurements of the given input set and uses the noisy measurements to construct an empirical distribution which along with the structure on  $P$  is used estimate the number of defectives in given set with required confidence level. This test is able to guide the random walk in Inner Module by identifying nodes on the binary tree that contain at least one defective item.

We describe the detailed working of the components in the hierarchical order. For the description, the value of  $d$  is assumed to be known. The case for the unknown number of defectives can be handled with a minor modification described at the end of the section.

### 8.3.1 The Outer Module

Consider the graph  $\mathcal{G}$  where each vertex corresponds to a subset of  $\mathcal{S}$  and the edges are given by the subset relation, i.e., there exists an edge between  $\mathcal{A}$  and  $\mathcal{B}$  for  $\mathcal{A}, \mathcal{B} \subset \mathcal{S}$  if and only if  $\mathcal{A} \subset \mathcal{B}$  or  $\mathcal{B} \subset \mathcal{A}$ . We use the term node and its corresponding subset interchangeably. We also adopt the following nomenclature to describe the nodes of  $\mathcal{G}$ . If  $\mathcal{A} \subset \mathcal{B}$ , we refer to  $\mathcal{B}$  and  $\mathcal{A}$  as parent and child nodes respectively. All nodes with a cardinality greater than  $d$  are referred to as internal nodes while ones with cardinality equal to  $d$  are called as leaf nodes. The nodes  $\mathcal{D}$  and  $\mathcal{S}$  are referred to as the target and the root nodes respectively.

The Outer Module involves inducing a biased random walk on this graph

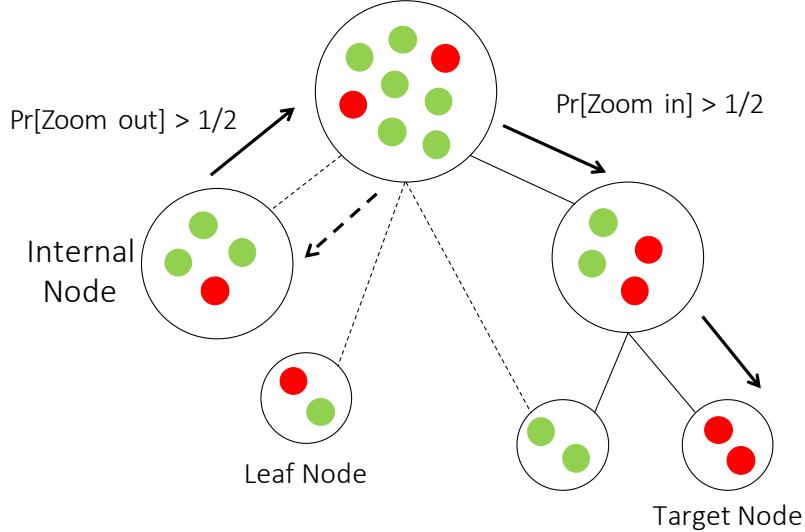


Figure 8.1: This is a representative figure for the outer module on a population of  $n = 8$  items that contain  $d = 2$  defective items (shown in red). The random walk begins at the root node. In the first step, it errs and zooms into the left child with 4 items (shown by a dotted arrow). Realizing the current node is not a parent of  $\mathcal{D}$ , it zooms back to its parent (here,  $\mathcal{S}$ ). In the next step, it correctly zooms into the right child followed by zooming into the target node in the step. It confirms that it is indeed the target node using a leaf level local sequential test.

$\mathcal{G}$ . The random walk is initialized at the root node. When the random walk is at any internal node (a subset  $\mathcal{A}$  of size  $m > d$ ), the outer module uses the inner module to obtain an estimate of the number of defective items in  $\mathcal{A}$  and a set,  $\mathcal{A}' \subset \mathcal{A}$ , of size at most  $m/d + d$  that contains all the defective items in  $\mathcal{A}$ . If the estimate equals  $d$ , the random walk zooms into the child,  $\mathcal{A}'$ , returned by the inner module. Otherwise, it zooms out to the parent of current node that was most recently visited in the sample path of the random walk. By zooming into subset that contains the  $d$  defective items, the random walk aims to continuously move to the parents of  $\mathcal{D}$  with smaller cardinality in the hope of arriving at the target node. At any internal node, the confidence of the estimate from the inner module is set to  $1 - p_{out}$ , where  $p_{out}$  is just required to be less than  $1/2$ . This ensures the required bias towards the parents of the target node.

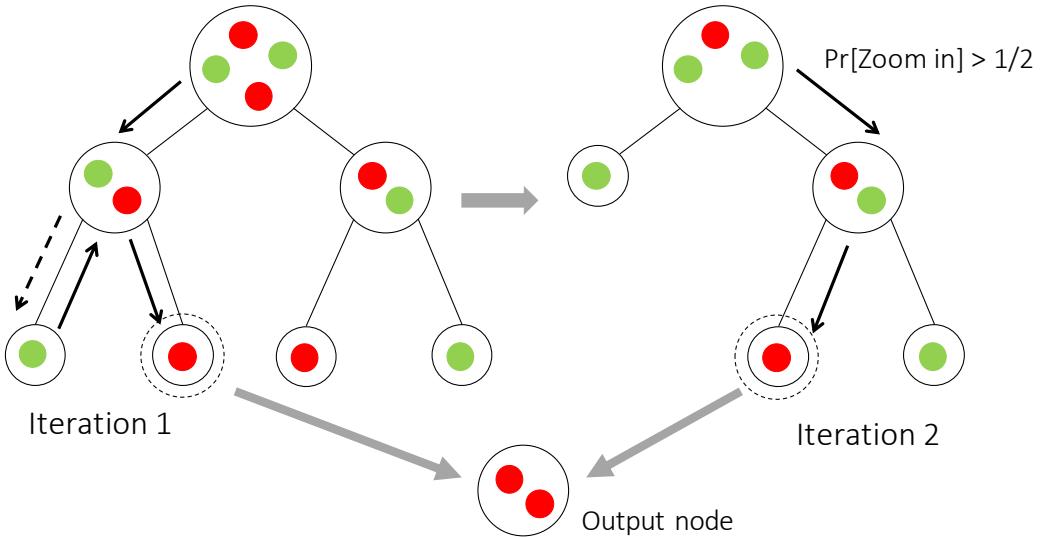


Figure 8.2: The above figure explains the working of the inner module on a set with  $m = 4$  items and  $d = 2$  defectives (shown in red). The left and right trees represent the first and the second iterations of the random walk respectively. In the first iteration, the random walk zooms into the correct child, makes a mistake, rectifies the same and identifies a defective item correctly. In the second iteration, it does not consider the identified item and carries out another random walk to identify the other defective item. Once both the defectives are identified, they are combined to output the final set  $\mathcal{A}'$ .

Once the random walk arrives at a leaf node, the outer module carries out the local sequential test on this leaf node. The sequential test queries the leaf node repeatedly until it obtains an estimate of the number of defective items contained in the set with the required confidence level. For the sequential test at the leaf node, the error probability for  $d$  defectives is set to  $\epsilon_1 = \frac{\epsilon(1-\exp(-2(p_{out}-1/2)^2)}{(\log_d n+1)}$  while it is set to  $p_{out}$  for all other cases. This implies that if the estimate returned by the sequential test equals  $d$  then the current leaf node contains exactly  $d$  defective items (i.e., it is the target node) with a probability of at least  $1 - \epsilon_1$  in which case the random walk will terminate and declare the current leaf node as the target node. In any other case, the obtained estimate of number of defective items has a confidence of just  $1 - p_{out} > 1/2$  and the random walk will zoom back to the parent of the current node in the same way as in the case of an

internal node since the current node is not a target node. This difference in confidence levels is crucial to obtain the order-optimal performance with respect to the overall probability of error.

### 8.3.2 The Inner Module

The Inner Module lies at the middle level of hierarchy of the algorithm and is used to guide the outer module. It is designed to take a set  $\mathcal{A} \subseteq \mathcal{S}$  of  $m$  items and an error probability  $\delta \in (0, 1)$  as inputs and output a set  $\mathcal{A}' \subset \mathcal{A}$  such that it contains all the defective items contained in  $\mathcal{A}$  with probability at least  $1 - \delta$  and satisfies  $|\mathcal{A}'| \leq m/d + d$ . It also returns a  $1 - \delta$  confidence estimate of the number of defectives in  $\mathcal{A}$ .

The inner module involves inducing a biased random walk on a binary tree constructed as follows. The  $m$  items in  $\mathcal{A}$  are first divided into  $d^2$  groups. Each group is referred to as a super item and contains roughly the same number of items (i.e.,  $\lfloor \frac{m}{d^2} \rfloor$  or  $\lceil \frac{m}{d^2} \rceil$ ). If  $m < d^2$ , each super item effectively reduces to a single item. The binary tree is obtained by successively splitting this set of  $d^2$  super items into equal halves until each leaf node corresponds to a single super item. In a similar vein to the outer module, we refer to the entire set as the root node, the non-leaf nodes as internal nodes and the super items that contain at least one defective item as target nodes. The objective of inner module is to identify all the target nodes.

The inner module involves a sequence of multiple random walks that identify the target nodes one by one. A random walk begins at the root node. At any internal node, it performs the local sequential test on both its children. Each

local test gives an estimate the number of defectives in the corresponding child with a confidence of  $1 - p_{in}$ , where  $p_{in}$  is set to any value less than 1/2. If exactly one child indicates the presence of a defective item, then the random walk zooms into that child. If both the children are estimated to contain a defective item, the random walk chooses a child uniformly at random and zooms into that child. Otherwise, it zooms back to its parent (with the parent of the root defined as itself).

Once the random walk reaches a leaf node, the inner module carries out a two sided local sequential test on the corresponding super item. The error probability corresponding to a result of 0 defective items is set to  $p_{in}$  while it is set to  $\delta_1 = \frac{p_{out}(1-\exp(-2(p_{in}-1/2)^2))}{d+1}$  for all other cases. If the result of this local test is 0, the random walk moves back to the parent. Otherwise, the random walk is terminated, the current super item is declared as target node and the corresponding number of defectives are noted.

Once a target node is identified, a new iteration of the random walk is initialized to identify the other target nodes. However, once a node is identified as a target node by a random walk, it is not considered in future random walks to avoid repetition. In particular, if  $v$  is a super item identified as a target node by the inner module and  $\mathcal{A}_v$  denotes an ancestor of  $v$ , then  $\mathcal{A}_v$  is updated to  $\mathcal{A}_v \setminus v$  for all random walks after  $v$  is identified. Additionally, to accommodate the fact the total number of target nodes are unknown, an additional local test needs to be carried out on the root node before testing its children. This test, referred to as termination test, involves carrying out a two sided local sequential test at the root node where the confidence associated with a result of 0 is set to  $1 - \delta_1$  and otherwise to  $1 - p_{in}$ . This termination test is to check the presence of target nodes

in the tree. The inner module terminates when the termination test returns a 0 or a total of  $d$  random walks have been carried out.

### 8.3.3 Confidence Based Local Sequential Test

In this section, we describe the local sequential test that guides the inner module. The test determines the number of defectives in a particular set of items, say  $\mathcal{A}$ , by sequentially querying it. Let  $Y_t \in \mathcal{Y}$  denote the (noisy) result of the test at time instant  $t$ . For each time  $t$ , we compute the vector  $(\hat{p}_0(t), \hat{p}_1(t), \dots, \hat{p}_K(t))$  where  $\hat{p}_i(t) = \frac{1}{t} \sum_{s=1}^t \mathbb{1}\{Y_s = i\}$  for all  $i = \{0, 1, \dots, K\}$ . Let  $\hat{i}_t = \arg \max_i \hat{p}_i(t)$  and  $\hat{j}_t = \arg \max_{j \neq \hat{i}_t} \hat{p}_j(t)$ . The local test begins with querying  $\mathcal{A}$ . Starting at  $t = 1$ , it checks the following condition:

$$\hat{p}_{\hat{i}_t}(t) - \hat{p}_{\hat{j}_t}(t) > \sqrt{\frac{5}{t} \log \left( \frac{6 \log t}{\sqrt{q}} \right)}. \quad (8.3)$$

If it is satisfied, the local test terminates and outputs  $\hat{i}_t$ . Otherwise, another measurement of  $\mathcal{A}$  is taken and the process is repeated. The motivation behind the test is to use the structure on  $P$  which ensures that the true value is the most likely outcome of the test. The above routine precisely estimates the most likely outcome and uses its gap with the next most likely outcome to ensure the required level of confidence.

For the case of unknown  $d$ , we can use the local sequential test to first estimate the value of  $d$  with confidence  $1 - \epsilon/2$  and run the above algorithm with an error budget of  $\epsilon/2$ .

## 8.4 Analysis

The following theorem shows that the number of tests required in HRW-QGT are  $O(\log n + \log(1/\epsilon))$ , thereby establishing the order optimality of the proposed strategy.

**Theorem 8.4.1.** *Consider the HRW-QGT algorithm where the outer and inner modules are run with parameters  $p_{out}$  and  $p_{in}$  respectively and  $\epsilon$  is the allowed probability of error. If  $\tau_{in}(d, p_{in}, p_{out})$  and  $\tau_{HRW}(d, n, \epsilon)$  indicate the random number of tests carried out during the inner module and the complete algorithm respectively, then we have,*

$$\begin{aligned} \mathbb{E}[\tau_{in}(d, p_{in}, p_{out})] &\leq \kappa \left( \frac{40}{\rho^2} \log \left( \frac{12}{\sqrt{p_{in}}} \log \left( \frac{240/\rho^2}{\sqrt{p_{in}}} \right) \right) + 2 \right) \\ &+ d \left( \frac{40}{\rho^2} \log \left( 12 \sqrt{\frac{\kappa}{2p_{out}}} \log \left( \frac{240}{\rho^2} \sqrt{\frac{\kappa}{2p_{out}}} \right) \right) + 2 \right) \end{aligned} \quad (8.4)$$

$$\begin{aligned} \mathbb{E}[\tau_{HRW}(d, n, \epsilon)] &\leq \gamma (\log_d(n) + 1) \mathbb{E}[\tau_{in}(d, p_{in}, p_{out})] \\ &+ \frac{40}{\rho^2} \log \left( \frac{12}{\sqrt{\epsilon_1}} \log \left( \frac{240}{\sqrt{\epsilon_1}\rho^2} \right) \right) + 2 \end{aligned} \quad (8.5)$$

where  $\beta = (1 - \exp(-2(p_{in} - 0.5)^2))^{-1}$ ,  $\kappa = 4d\beta \log d$ ,  $\gamma = (1 - \exp(-2(p_{out} - 0.5)^2))^{-1}$  and  $\epsilon_1 = \epsilon / (\gamma(\log_d n + 1))$ .

*Proof.* The proof of the theorem is based on the following two lemmas that characterize the performance of the local test and one iteration of random walk. Please refer to Appendix G for details.

**Lemma 8.4.2.** *Consider the local sequential test described in Section 8.3.3 being carried out on a set  $\mathcal{A}$ . If  $\tau_{sq-test}$  denotes the random number of measurements of  $\mathcal{A}$  taken to obtain an estimate with a confidence level  $1 - q$ , then we have,*

$$\mathbb{E}[\tau_{sq-test}] \leq \frac{40}{\rho^2} \log \left( \frac{12}{\sqrt{q}} \log \left( \frac{240}{\sqrt{q}\rho^2} \right) \right) + 2. \quad (8.6)$$

where  $q$  is the allowed probability of error. The probability of error in the sequential procedure can be bounded as

$$\Pr(\hat{i}_{\tau_{\text{sq-test}}} \neq x^*) \leq q. \quad (8.7)$$

**Lemma 8.4.3.** Consider the routine of a biased random walk on a tree with depth  $h$  guided by a confidence-based test that takes  $N_0(\delta)$  samples to predict the result of the test with a confidence of  $1 - \delta$ . If  $\tau_{RWT}(p, \epsilon)$  denotes the number of samples taken during a random walk with a bias of  $1 - p > 1/2$  and a target accuracy of  $1 - \epsilon$ , then we have

$$\mathbb{E}[\tau_{RWT}(p, \epsilon)] \leq 2\beta(h + 1)N_0(p) + N_0(\epsilon') \quad (8.8)$$

where  $\beta = (1 - \exp(-2(p - 0.5)^2))^{-1}$  and  $\epsilon' = \epsilon/(\beta(h + 1))$ .

Using the Lemma 8.4.3 repeatedly for  $d$  iterations with  $N_0(\delta)$  obtained from Lemma 8.4.2, we obtain the bound on the samples taken by the inner routine. That result is again combined with Lemma 8.4.3 to obtain the performance of HRW-QGT.

□

CHAPTER 9

**SEQUENTIAL ALGORITHM FOR TESTING UNIFORMITY OF  
DISTRIBUTION**

## 9.1 Introduction

Consider the following composite hypothesis testing problem: Given samples of a random variable with density function  $f$ , we aim to determine whether  $f$  is the uniform distribution  $u$  over  $[0, 1]$  (the null hypothesis) or  $f$  is at least  $\varepsilon$  distance away (in  $\ell_1$ ) from the uniform distribution. The objective is to minimize the sample complexity subject to the constraint of both the Type I and Type II errors being capped below a prescribed value  $\delta \in (0, 1)$ .

It turns out that without imposing structural constraints on the set of alternative distributions, the above hypothesis testing problem is not testable: no algorithm can achieve diminishing error probability as the number of samples increases [179, 7]. One class of distributions that are testable is the class of monotone distributions [7, 6], for which it has been shown that the sample complexity is of the order  $O(1/\varepsilon^2)$ . General conditions on testability remain unknown, and uniformity testing of continuous distributions has not been well explored.

In contrast, there is an extensive literature on uniformity testing of discrete distributions with a support size  $m$ . The problem dates back to the so-called empty-box problem first posed by David in [86] and later generalized by Viktorova and Chistyakov in [310]. David cast the problem as throwing balls in  $m$  boxes and proposed to use the number of empty boxes or the number of boxes containing exactly one ball (a.k.a. the coincidence number) as test statistics for

uniformity testing<sup>1</sup>. More in-depth studies of using the coincidence test statistic (i.e., the number of letters in the distribution alphabet that see exactly one sample) for uniformity testing were given by Paninski in [220] and Huang and Meyn [142]. Specifically, Paninski showed that the sample complexity of the coincidence test is of  $O(\sqrt{m}/\varepsilon^2)$  under the condition that  $\varepsilon = \Omega(m^{-1/4})$ . Paninski also established that the sample complexity of the coincidence test is order optimal by providing a matching lower bound. Other test statistics used for uniformity testing include empirical  $\ell_2$  distance [28, 29, 116, 63], empirical  $\ell_1$  distance [93], and modified  $\chi^2$ -statistic [5].

Most existing algorithms for uniformity testing are batch algorithms in the sense that all samples are collected prior to performing the test, which makes it necessary to focus on the most challenging alternative distributions (i.e., those that are at the minimum distance  $\varepsilon$  away from the uniform). While such approaches are sufficient to obtain minimax-optimal sample complexity, they result in significantly suboptimal sample complexity for almost all instances in the class of alternate distributions. Such suboptimality may have significant implications in practice. For example, when the alternative hypotheses represent anomalies in critical infrastructure, it is often more crucial to detect quickly those severe anomalies far away from the normal state<sup>2</sup>, as more severe anomalies often carry more risk. It is therefore highly desirable to have a test adaptive to the underlying alternative distribution, which motivates the sequential test developed in this work.

---

<sup>1</sup>The work by [86] considered the problem of testing whether samples are drawn i.i.d. from a known continuous distribution. The problem was reduced to a discrete problem by quantizing samples into  $m$  bins without discussing the choice of  $m$ . The heuristic approaches he proposed are for solving the discrete problem.

<sup>2</sup>In most applications, the probabilistic model of the normal state is known, which can be transformed to a uniform distribution. Anomaly detection can then be cast as uniformity testing.

### 9.1.1 Main results

We consider uniformity testing of Lipschitz continuous (density) distributions, which is arguably more general than the class of monotone distributions studied in [7] and [6]. Another theme that separates this work from existing literature is the sequential aspect of the proposed tests that adapt to the underlying alternative distribution.

Referred to as the Adaptive Binning Coincidence (ABC) test, the proposed strategy adapts to the unknown alternative distribution in two ways. First, it partitions the set of alternative distributions into layers based on their distances to the uniform distribution. It then sequentially eliminates the alternative distributions layer by layer in decreasing distance to the uniform, and subsequently takes advantage of favorable situations of a distant alternative by exiting early. Second, it adapts, across layers of the alternative distributions, the resolution level of the discretization for computing the coincidence statistic. The farther away the layer is from the uniform, the coarser the discretization is needed for eliminating/exiting this layer. It thus exits both *early* in the detection process and *quickly* by using a lower resolution to take advantage of favorable alternative distributions. We establish the sample complexity of the ABC test, which characterizes its adaptivity.

The ABC test builds on an adaptive sequential coincidence test for discrete distributions, which is of independent interest. Due to the adaptivity, this sequential test improves the sample complexity under the alternative hypothesis from  $O(\sqrt{m}/\varepsilon^2)$  of Paninski's batch algorithm to  $O(\gamma^{-2} \sqrt{m} \log(1/\gamma))$ , where  $\gamma$  is the distance of the underlying alternative distribution to the uniform and is greater than the minimum distance  $\varepsilon$ . This demonstrates that the sequential

coincidence test adapts to the *realized* distance  $\gamma$  in an optimal order (up to a logarithmic factor) in terms of sample complexity.

### 9.1.2 Related Work

The problem of estimating and testing properties of an unknown distribution has an extensive literature (see an excellent survey in [56]). The problem of uniformity testing is among the most widely studied problems among this class. As discussed earlier, most existing work focuses on discrete distributions and batch algorithms (see [220, 142, 5]). A notable exception to these batch-based strategies is work by Batu and Canonne in [27]. The test developed in [27] uses 2-way and 3-way collisions among the set of samples as test statistics, which in expectation correspond to the  $\ell_2$  and  $\ell_3$  norms of the distribution. The sample complexity of this algorithm, while having a distribution dependent term of  $\ell_3$  norm of the distribution, maintains a term determined by the worst distance to the alternative set,  $\varepsilon$ . In other words, the adaptivity to the underlying hypothesis is only *partial*, and the second term in terms of the worst distance  $\varepsilon$  may dominate. The test strategies proposed here, however, are *fully* adaptive as their sample complexities depend only on the  $\ell_1$  distance of the underlying distribution  $p$  to the uniform distribution, with no dependence on  $\varepsilon$ . In a concurrent work, Fawzi *et al.* [107] also proposed a sequential extension of the batch-based algorithm for testing closeness of distributions. Their results also establish the benefit of adaptivity obtained by sequential testing as opposed to the batch based approach. However, their work is limited to testing closeness of discrete distribution while our focus is on continuous distributions.

The literature on uniformity testing of continuous distributions is slim. As mentioned previously, it is necessary to impose a certain structure on the class of distributions being considered in order to ensure feasibility of the problem. For the case when the underlying distribution is monotone, Adamaszek *et al.* [7] and Acharya *et al.* [6] have proposed algorithms that offer optimal sample complexities using sample mean and modified  $\chi^2$  test as test statistics respectively. Diakonikolas *et al.* [95] also studied identity testing of a distribution under various structural assumptions. In [146], Ingster studied uniformity testing against an alternative class of smooth densities, which includes the Lipschitz continuous distributions studied here. The key differences are that the algorithm in [146] uses the  $\chi^2$  test statistic and does not offer adaptivity to the underlying distribution. The focus of this work is to develop fully adaptive test strategies employing the much simpler test statistic of coincidence number.

## 9.2 Uniformity Testing of Discrete Distributions

In this section, we consider uniformity testing of discrete distributions and develop a sequential test employing the coincidence statistic. The results obtained for the discrete problem form the foundation for tackling the continuous problem in the next section.

The key property of this sequential coincidence test (SCT) is that it adapts to the unknown alternative distribution in the composite set. More specifically, the sample complexity of SCT under the alternative hypothesis scales optimally with respect to the distance  $\gamma$  of the realized alternative distribution to the uniform. This is in sharp contrast to batch algorithms whose sample complexity

is determined by the worst-case alternative distribution seeing the minimum distance  $\varepsilon$  to the uniform.

### 9.2.1 Problem Formulation

Consider a binary hypothesis testing problem where the null hypothesis  $H_0$  is the uniform distribution  $u$  with a support size of  $m$ . Without loss of generality, we assume that the support set is  $\{1, 2, \dots, m\}$  denoted by  $[m]$ . The alternative hypothesis  $H_1$  is composite: it consists of all distributions over  $[m]$  whose  $\ell_1$  distance to  $u$  is no smaller than  $\varepsilon$ . More specifically, let  $C(\varepsilon)$  denote the composite set of alternative distributions under  $H_1$ , we have

$$C(\varepsilon) = \{q \in \mathcal{P}([m]) : \|q - u\|_1 > \varepsilon\},$$

where  $\mathcal{P}([m])$  denotes all distributions over  $[m]$ ,  $\|q - u\|_1 = \sum_{i=1}^m |q_i - 1/m|$  is the  $\ell_1$  distance between distribution  $q$  and the uniform distribution  $u$ .

For the hypothesis testing problems, i.i.d. samples are drawn from either  $u$  (if  $H_0$  is true) or a specific distribution in  $C(\varepsilon)$  (if  $H_1$  is true), unknown to the decision maker. The goal is to determine, based on the random samples, which hypothesis is true. The probabilities of false alarm and miss detection need to be capped below a given  $\delta$  ( $\delta \in (0, 1)$ ) for all alternative distributions in  $C(\varepsilon)$ .

### 9.2.2 Sequential Coincidence Test

Existing work on uniformity testing all focuses on batch methods (a.k.a. the fixed-sample-size tests). Specifically, based on the reliability constraint  $\delta$  and the

minimum separation  $\varepsilon$  between  $H_0$  and  $H_1$ , the number of required samples is pre-determined to ensure  $\delta$ -reliability in the worst case of a closest (i.e., distance  $\varepsilon$ ) alternative.

We propose a sequential test SCT that adapts to the unknown alternative. When the alternative is at a distance greater than  $\varepsilon$  from  $u$ , the sequential test takes advantage of the favorable situation and exits towards  $H_1$  with fewer samples. In particular, the sample complexity of SCT scales optimally with the *realized* distance  $\gamma$  rather than the minimum distance  $\varepsilon$ .

SCT employs the coincidence statistic. For a give set of samples  $\mathcal{S}$ , the coincidence  $K_1(\mathcal{S})$  is the number of symbols in  $[m]$  that appear exactly once in  $\mathcal{S}$ . Specifically, let  $n_j$  denote the number of appearances of symbol  $j$  in  $\mathcal{S}$ . Then

$$K_1(\mathcal{S}) = \sum_{j=1}^m \mathbb{1}\{n_j = 1\},$$

where  $\mathbb{1}\{\cdot\}$  is the indicator function. Let  $K_1(n)$  denote the coincidence number of  $n$  i.i.d. samples drawn from a given distribution  $p$ . It is a random variable whose distribution is determined by  $n$  and  $p$ . We then introduce the constant  $c_u(n)$ , which is the expected value of  $K_1(n)$  under the uniform distribution:

$$c_u(n) = \mathbb{E}_u[K_1(\mathcal{S})], \quad \text{where } |\mathcal{S}| = n, \quad \mathcal{S} \stackrel{\text{i.i.d.}}{\sim} u.$$

SCT proceeds in epochs. In each epoch the test takes  $\Theta(\sqrt{n})$  additional samples. At the end of each epoch, based on all the samples  $\mathcal{S}$  collected so far, the algorithm decides whether there is sufficient evidence to exit towards  $H_1$ . This decision is made by comparing the difference between  $K_1(\mathcal{S})$  and  $c_u(|\mathcal{S}|)$  to a carefully chosen threshold. If the difference exceeds the threshold (indicating a sufficient separation between the coincidence number of the samples  $\mathcal{S}$  and the

expected coincidence number of the uniform), the algorithm terminates and declares  $H_1$ . Otherwise, the algorithm enters the next epoch. In the event that the process reaches the maximum number  $\Theta(\varepsilon^{-2})$  of epochs without exiting towards  $H_1$ , the algorithm terminates and declares  $H_0$ . A pseudo code for the algorithm is given in Algorithm 20.

---

**Algorithm 20** Sequential Coincidence Test (SCT)

---

```

Input:  $m, \varepsilon, \delta \in (0, 1)$ 
Set  $k \leftarrow 1, t \leftarrow 0, \kappa = 112\varepsilon^{-2}, \mathcal{S} = \{\}$ 
while  $k \leq \kappa$  do
     $n_k \leftarrow \lceil k \sqrt{m \log(k + 2/\delta)} \rceil$ 
     $\tau_k \leftarrow 7n_k \sqrt{\frac{\log(k + 2/\delta)}{m}}$ 
    repeat
        Obtain a sample  $X_t$ 
         $\mathcal{S} \leftarrow \mathcal{S} \cup X_t$ 
         $t \leftarrow t + 1$ 
    until  $t == n_k$ 
    if  $Z_k := c_u(n_k) - K_1(\mathcal{S}) > \tau_k$  then
        Output  $\leftarrow H_1$ 
        break
    end if
     $k \leftarrow k + 1$ 
end while
if  $k > \kappa$  then
    Output  $\leftarrow H_0$ 
end if
return Output

```

---

The sequential detection process of SCT can be visualized as peeling an onion: the core of the onion is the uniform distribution and its  $\varepsilon$ -neighbors; the layers represent alternative distributions at increasing distance to the uniform distribution<sup>3</sup>. Each epoch peels one layer of the onion, either by exiting towards  $H_1$  (if the realized alternative distribution resides in this layer) or by eliminat-

<sup>3</sup>The epoch structure of SCT effectively quantizes the distance to  $u$ , hence forming a finite partition of the alternative distributions in  $C(\varepsilon)$ . More specifically,  $C(\varepsilon)$  is partitioned into  $\kappa$  layers, where  $\kappa = 112\varepsilon^{-2}$  is the maximum epoch number defined in Algorithm 20. Each epoch peels off one layer.

ing this set of alternative distributions and moving to the next layer closer to the core. If all outer layers are eliminated (i.e., all alternative distributions in  $C(\varepsilon)$  are eliminated), SCT terminates and declares  $H_0$ . The ability of peeling the onion layer by layer is rooted in the fact that when the samples  $\mathcal{S}$  are drawn from a distribution  $\gamma$ -distance away from  $u$ , the difference in coincidence numbers  $c_u(n_k) - K_1(\mathcal{S})$  scales proportionally with  $\gamma^2$ . This difference hence exceeds the threshold early when  $\gamma$  is large (i.e., when the alternative distribution resides farther from the core of the onion).

### 9.2.3 Sample Complexity

The expected sample complexity of SCT is characterized in the following theorem.

**Theorem 9.2.1.** *For  $m > m_0$  and  $\varepsilon = \Omega(m^{-1/8})$ , where  $m_0 = \min\{l : 1123l^{1/4}e^{-0.25\sqrt{l}} \leq \delta\varepsilon^2\}$ , we have*

- Under  $H_1$  with an alternative distribution  $p$  that is  $\gamma$  away from  $u$ , the expected sample complexity of SCT is  $O\left((\sqrt{m}/\gamma^2)\sqrt{\log(1/\gamma + 1/\delta)}\right)$
- Under  $H_0$ , the expected sample complexity of SCT is  $O\left((\sqrt{m}/\varepsilon^2)\sqrt{\log(1/\varepsilon + 1/\delta)}\right)$
- Under both  $H_1$  and  $H_0$ , the probability of correct detection under SCT is at least  $1 - \delta$ .

As evident from the above theorem, the sample complexity of SCT under  $H_1$  adapts to the distance  $\gamma$  of the alternative distribution  $p$  to the uniform distribution. Since  $p \in C(\varepsilon)$ , we have  $\gamma > \varepsilon$ , implying that the sample complexity is smaller than the fixed-sample-size approaches which offer a sample complexity

of  $O(\varepsilon^{-2} \sqrt{m})$ . Moreover, the adaptivity of SCT to the realized distance  $\gamma$  is order-optimal (up to a logarithmic factor). This can be shown by noting that even with the knowledge of  $\gamma$ , the lower bound given by Paninski dictates  $\Omega(\gamma^{-2} \sqrt{m})$  samples to ensure a constant probability of reliability.

Furthermore, in addition to the near-optimal scaling with  $\gamma$ , SCT offers better scaling of sample complexity with respect to  $\delta$ , the error probability, as compared to the batch algorithms. In particular, the batch algorithms are designed to guarantee a certain constant probability of error, and the common technique to extend such tests to guarantee an arbitrary probability of error  $\delta$  is to repeat the test sufficiently many times so that the result declared by the majority vote meets the confidence requirements. Such an approach results in a  $\log(1/\delta)$  dependence of sample complexity on  $\delta$  as opposed to the  $\sqrt{\log(1/\delta)}$  offered by SCT. Thus, the refined analysis required to analyze the sequential coincidence test not only demonstrates adaptivity to the underlying distribution but also results in improved dependence on  $\delta$ . Such an improved dependence on  $\delta$  is also obtained for the empirical  $\ell_1$  distance based statistic proposed in the recent work by Diakonikolas *et al.* [93] but has not been established for other popular test statistics like the coincidence test statistic [220], modified  $\chi^2$ -based test [5] and empirical  $\ell_2$  distance based statistic [28, 29, 116, 63].

In addition to the dependence on  $\gamma$  and  $\delta$ , we would also like to briefly mention the regime of input parameters  $m$  and  $\varepsilon$  for which this result holds. The lower bound  $m_0$  is required to ensure the support size is large enough to ensure a confidence of  $\delta$  in the sparse regime. Moreover, the regime of  $\varepsilon$  for which this result is applicable can also in part be attributed to the fact that the coincidence statistic works well only in the sparse regime. We believe that the  $\varepsilon = \Omega(m^{-1/8})$

requirement as opposed to the standard requirement of  $\varepsilon = \Omega(m^{-1/4})$  for sparse regime is merely an analysis artifact and can be improved using better techniques for bounding the moment generating function of  $K_1$ . Please refer to Appendix H.1 for a detailed proof of the Theorem.

## 9.3 Uniformity Testing of Continuous Distributions

### 9.3.1 Problem Formulation

We now consider uniformity testing of continuous distributions, particularly Lipschitz continuous distributions with bounded support. Specifically, the null hypothesis  $H_0$  is the uniform distribution  $u$  over  $[0, 1]$ . The alternative composite hypothesis  $H_1$  is the set of  $L$ -Lipschitz distributions whose  $\ell_1$  distance to  $u$  is no smaller than  $\varepsilon$ . Let  $\mathcal{P}([0, 1]; L)$  denote the set of distributions over  $[0, 1]$  that are absolutely continuous with the Lebesgue measure on  $[0, 1]$  and whose density functions are  $L$ -Lipschitz. Specifically, for all distributions  $q \in \mathcal{P}([0, 1]; L)$ , the density functions  $f_q(x)$  satisfies, for all  $x, y \in [0, 1]$ ,

$$|f_q(x) - f_q(y)| \leq L|x - y|.$$

The composite set of alternative distributions under  $H_1$  is given by

$$C_L(\varepsilon) = \left\{ q \in \mathcal{P}([0, 1]; L) : \int_0^1 |f_q(x) - 1| dx > \varepsilon \right\}.$$

The objective of the uniformity testing is the same as in the discrete problem: to determine, with a reliability constraint of  $\delta$ , whether the observed random samples are generated from  $u$  ( $H_0$ ) or from a distribution in  $C_L(\varepsilon)$  ( $H_1$ ).

### 9.3.2 Adaptive Binning Coincidence Test

We now generalize SCT to the continuous problem specified above. Our goal is to preserve the adaptivity of SCT to the underlying alternative distribution under  $H_1$ .

The relation among the set of continuous distributions can still be visualized as an onion:  $u$  and its  $\varepsilon$ -neighbors form the core, and alternative distributions in  $C_L(\varepsilon)$  form layers according to their distances to  $u$ . The algorithm still aims to determine, sequentially over epochs, which layer the underlying distribution of the observed samples resides, starting from the outer-most and moving towards the core. The key question in the continuous problem is how to infer, from a coincidence type of statistic, whether the underlying distribution resides in the current layer. A straightforward answer to this question is discretization: a uniform binning of the support set  $[0, 1]$  with coincidence number defined with respect to the bin labels of the random samples. Much less obvious is the choice of the resolution level for the discretization, i.e., how finely to bin the continuum domain. A key rationale behind the proposed ABC (Adaptive Binning Coincidence) test is that the farther away the layer is from the core, the coarser the discretization is needed for inferring whether the underlying distribution resides in this layer. More specifically, not only the test can exit early when the realized distance  $\gamma$  is favorable, but also the number of required samples for making the peeling decision is fewer due to a coarser discretization. In other words, ABC adapts to the unknown realized distance  $\gamma$  by exiting both *early* in the detection process and *quickly* by using a lower resolution to take advantage of favorable alternative distributions.

We can now describe the ABC test, which proceeds in a similar epoch struc-

ture as SCT with two key modifications. First, the resolution of the discretization increases at a carefully chosen rate over epochs, and the number of samples taken in each epoch is adjusted accordingly. Specifically, let  $m_k$  denote the number of discretization bins in epoch  $k$ , where  $m_k$  increases at the rate of  $k^4 \log k$ . The number of samples taken in epoch  $k$  is  $\Omega(\sqrt{m_k})$ , which retains the same squared-root relation to the effective support size  $m_k$  as in the discrete case. Second, the coincidence number  $K_1(S)$  in each epoch is computed by rebinning all observed samples (including those obtained in previous epochs) based on the refined discretization  $m_k$  in the current epoch. A detailed description of the algorithm is given in Algorithm 21, where  $K_1(S; m)$  denotes the coincidence number computed over the set  $S$  of samples when the interval is uniformly divided into  $m$  bins. Similarly,  $c_u(n; m)$  denotes the expected coincidence number of  $n$  samples from the uniform distribution with a support of  $m$  bins. The constant<sup>4</sup>  $c_0 \geq \max\{28212, m_0, 2L\}$ , where  $m_0$  is defined in Theorem 9.2.1.

We would like to point out that while we have described ABC for uniform distribution, it can be easily extended to cases where the null hypothesis corresponds to an arbitrary distribution in  $\mathcal{P}([0, 1]; L)$ . In this case, we can employ a non-uniform binning strategy that bins the null hypothesis into a discrete uniform distribution. It can be shown that the sample complexity as analyzed next holds for this more general problem provided that the null hypothesis distribution is lower bounded by a constant.

---

<sup>4</sup>The constant 28212 as a lower bound for  $c_0$  arises from the conditions imposed during analysis. Please refer to Appendix H.1 for exact expressions for these conditions. We would also like to point out that the constants are not optimized. The analysis focuses on the order.

---

**Algorithm 21** Adaptive Binning Coincidence (ABC) Test

---

```
Input:  $\varepsilon, L, \delta \in (0, 1)$ 
Set  $k \leftarrow 1, t \leftarrow 0, \kappa \leftarrow 576\varepsilon^{-2}, \mathcal{S} \leftarrow \{\}$ 
while  $k \leq \kappa$  do
     $m_k \leftarrow \lceil c_0 k^4 \log(k + 2/\delta) \rceil$ 
     $n_k \leftarrow \lceil \sqrt{c_0} k^3 \log(k + 2/\delta) \rceil$ 
     $\tau_k \leftarrow 9n_k \sqrt{\frac{\log(k + 2/\delta)}{m_k}}$ 
repeat
    Obtain a sample  $X_t$ 
     $\mathcal{S} \leftarrow \mathcal{S} \cup X_t$ 
     $t \leftarrow t + 1$ 
until  $t == n_k$ 
if  $Z_k := c_u(n_k; m_k) - K_1(\mathcal{S}; m_k) > \tau_k$  then
    Output  $\leftarrow H_1$ 
    break
end if
     $k \leftarrow k + 1$ 
end while
if  $k > \kappa$  then
    Output  $\leftarrow H_0$ 
end if
return Output
```

---

### 9.3.3 Sample Complexity

The theorem below establishes the sample complexity of ABC and its adaptivity to the realized distance  $\gamma$  under  $H_1$ .

**Theorem 9.3.1.** • Under  $H_1$  with an alternative distribution  $p$  that is  $\gamma$  away from  $u$ , the expected sample complexity of ABC is  $O\left(\gamma^{-6} \log(1/\gamma + 1/\delta)\right)$ .  
• Under  $H_0$ , the expected sample complexity of ABC is  $O\left(\varepsilon^{-6} \log(1/\varepsilon + 1/\delta)\right)$ .  
• Under both  $H_1$  and  $H_0$ , the probability of correct detection under ABC is at least  $1 - \delta$ .

The lower bound on the sample complexity for this problem is  $\Omega(\sqrt{L}\varepsilon^{-5/2})$  as

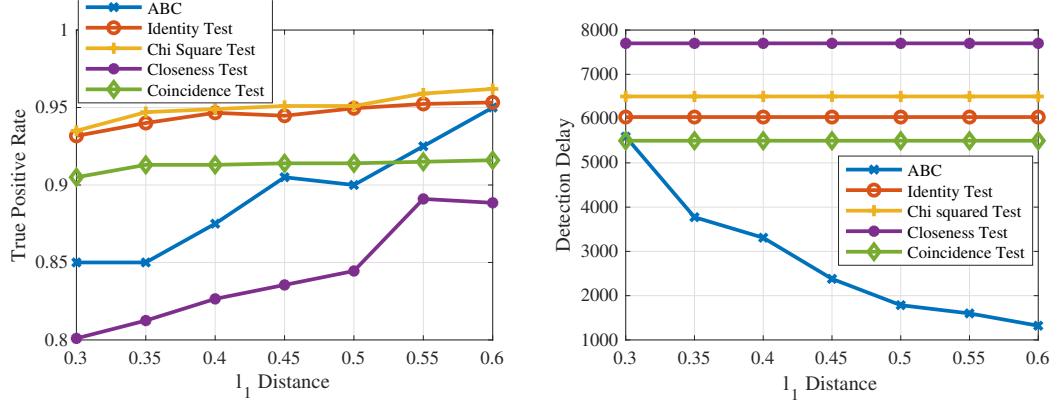


Figure 9.1: True Positive Rate (right) and Empirical Sample Complexity (left) vs  $\ell_1$  distance for discrete alternative distributions.

shown in [146]. Evidently, there is a significant gap between the lower bound on sample complexity and the sample complexity guarantees offered by ABC. This gap, we believe is rooted in that coincidence statistic is informative only in sparse regimes where the number of samples is of sublinear order of the support size. We conjecture that the gap to the lower bound is unavoidable for tests using the coincidence statistic. In other words, we conjecture that while coincidence statistic is sufficient for achieving order-optimal sample complexity in the discrete case, it ceases to remain so in the continuous case. An interesting question is to explore is a lower bound on the sample complexity achievable by the simple test statistic of coincidence number. It is also of interest to develop adaptive tests using more complex test statistics such as  $\chi^2$ . While the dependence on  $\varepsilon$  is suboptimal, we would like to point out that ABC achieves optimal scaling with the Lipschitz constant  $L$ . A detailed proof of the above Theorem can be found in Appendix H.2.

## 9.4 Simulations

We corroborate our theoretical findings by testing the algorithms empirically on synthetic and real-world datasets. We designed synthetic datasets to test the hypothesis testing methods for both continuous and discrete uniform distributions, which are described in Sections 9.3 & 9.2, respectively. Then we used measurements collected from a real-world signal-driven simulated system to demonstrate the performance of our adaptive detection method for real-world system.

For each of these datasets, we ran several the fixed sample size coincidence tests as baselines to our work, including the fixed-sample-size coincidence test by Paninski [220], the fixed sample size chi-squared test in Acharya *et al.* [5], the high probability identity test in Diakonikolas *et al.* [93], and the closeness test between two distributions proposed by Chan *et al.* [63]. These methods are denoted by coincidence test, chi-squared test, identity test and closeness test respectively.

We consider the problem of binary hypothesis testing described in Section 9.2 with  $m = 20000$  and  $\varepsilon = 0.3$ . In the experimental setup, we consider a set of distributions in  $C(\varepsilon)$  parametrized by  $\gamma$  and whose probability mass functions are given as follows:

$$p_{2i-1}^\gamma = (1 + \gamma)/m; p_{2i}^\gamma = (1 - \gamma)/m,$$

for  $i = \{1, 2, \dots, m/2\}$ . It can be noted that the  $\ell_1$  distance of  $p^\gamma$  from the uniform distribution is  $\gamma$ . We consider 7 distributions corresponding to the values of  $\gamma \in \{0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6\}$ . For all algorithms, their thresholds are set to obtain an false positive rate of at most 0.2. The constants in the threshold

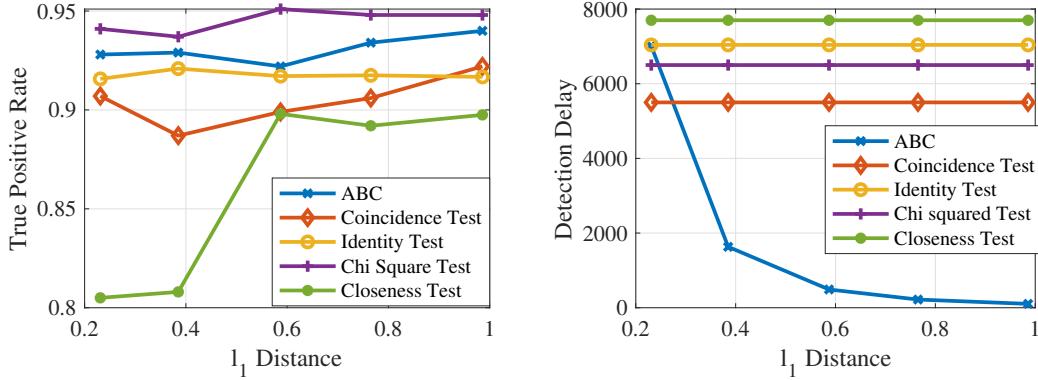


Figure 9.2: True Positive Rate (right) and Empirical Sample Complexity (left) vs  $\ell_1$  distance for continuous alternative distributions.

for both the algorithms are optimized using grid search to give the best empirical sample complexity. Fig. 9.1 shows the empirical sample complexities and false positive rates obtained under different distributions. For the ABC, the expected sample complexity is obtained by taking an average over 1000 Monte Carlo runs. As expected, the sample complexities for the fixed sample size testing techniques are the same for all the distributions. However, the sample complexity of the ABC adapts to the  $\ell_1$  distance of the underlying distribution and decreases as the  $\ell_1$  distance increases, demonstrating the benefit of the proposed approach. As shown by Fig. 9.1(right), we also see that the ABC achieves similar level of true positive rates with better empirical sample complexities compare with fixed sample-size methods.

We tested the continuous testing method described in Section 9.3 with beta distributions as alternatives. The parameters of beta distribution  $(\alpha, \beta)$  were taken in the set  $\{(1.5, 1.5), (2, 2), (3, 3), (3, 5), (3, 8)\}$ . Fig. 9.2 displays the empirical sample complexities and true positive rates obtained for all methods at false positive rate at most 0.2. Similar to the discrete case, ABC demonstrated its adaptive detection delay over fixed-sample size methods with similar level of

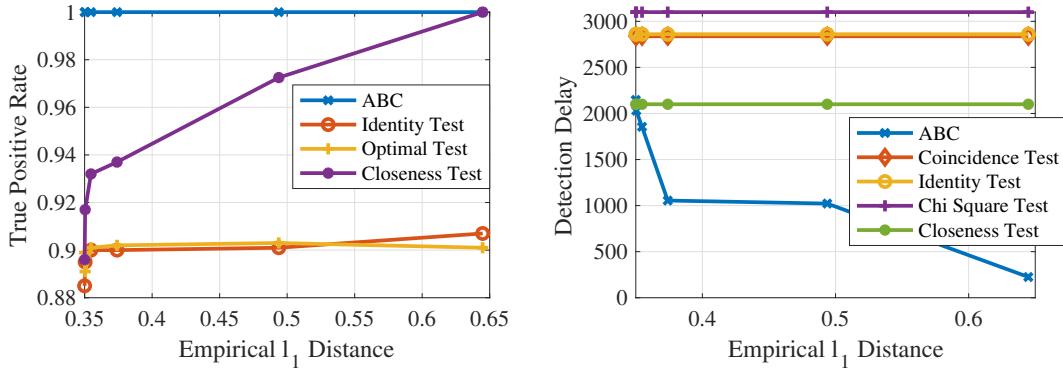


Figure 9.3: True Positive Rate (right) and Empricial Sample Complexity (left) vs  $\ell_1$  distance for alternative samples collected from simulated power system.

accuracy for all continuous alternative distributions.

To show the performance of ABC in real-world applications, we conducted testing on a current samples collected from a real power system provided by EPFL<sup>5</sup> [276]. To obtain alternative cases with different  $\ell_1$  distance, we constructed a synthetic system similar to the EPFL campus grid, and simulated the fault scenarios with different electrical distance. We then adopted method in Wang and Tong [319] to transform the waveform to samples on  $[0, 1]$ , and anomaly-free samples will be transformed to *i.i.d* samples of continuous uniform distribution. Finally, ABC and other testing method for uniform distributions were applied on the transformed sequence to detect anomalous and anomaly-free segments.

Fig. 9.3 shows the true positive rate and detection delay for the transformed power system data, collected at false positive rate 0.05. ABC is shown to have the optimal true positive rate and detection delay over other baselines, demonstrating its effectiveness in real-world applications.

<sup>5</sup><https://github.com/DESL-EPFL/Point-on-wave-Data-of-EPFL-campus-Distribution-Network>

## BIBLIOGRAPHY

- [1] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, 2011.
- [2] Y. Abbasi-Yadkori, D. Pal, and C. Szepesvari. Online-to-confidence-set conversions and application to sparse stochastic bandits. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics, AIS-TATS*, volume 22 of *Proceedings of Machine Learning Research*, pages 1–9, La Palma, Canary Islands, 2012. PMLR.
- [3] A. Abdi and F. Fekri. Quantized compressive sampling of stochastic gradients for efficient communication in distributed deep learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, AAAI*, volume 34, pages 3105–3112, 2020.
- [4] S. AbdulRahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. *IEEE Internet of Things Journal*, 8(7):5476–5497, 2021.
- [5] J. Acharya, C. Daskalakis, and G. Kamath. Optimal testing for properties of distributions. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*, pages 3591–3599, 2015.
- [6] J. Acharya, A. Jafarpour, A. Orlitsky, and A. T. Suresh. A competitive test for uniformity of monotone distributions. *Journal of Machine Learning Research*, 31:57–65, 2013.
- [7] M. Adamaszek, A. Czumaj, and C. Sohler. Testing monotone continuous distributions on high-dimensional real cubes. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 56–65. Association for Computing Machinery (ACM), 2010.
- [8] M. Agarwal, V. Aggarwal, and K. Azizzadenesheli. Multi-Agent Multi-Armed Bandits with Limited Communication. *Journal of Machine Learning Research*, 23, 2022.
- [9] M. Aigner. Search problems on graphs. *Discrete Applied Mathematics*, 14(3):215–230, 1986.

- [10] M. Aigner and M. Schughart. Determining defectives in a linear order. *Journal of Statistical Planning and Inference*, 12:359–368, 1985.
- [11] S. Amani, T. Lattimore, A. György, and L. F. Yang. Distributed Contextual Linear Bandits with Minimax Optimal Communication Cost, 2022.
- [12] M. Anthony and P. L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.
- [13] S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 32, 2019.
- [14] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.
- [15] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.
- [16] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science, FOCS*, pages 322–331, 1995.
- [17] P. Awasthi, M.-F. Balcan, N. Haghtalab, and R. Urner. Efficient learning of linear separators under bounded noise. In *Proceedings of the 28th Annual Conference on Learning Theory, COLT*, pages 167–190. PMLR, 2015.
- [18] P. Awasthi, M. F. Balcan, N. Haghtalab, and H. Zhang. Learning and 1-bit compressed sensing under asymmetric noise. In *Journal of Machine Learning Research*, volume 49, pages 152–192, 2016.
- [19] P. Awasthi, M. F. Balcan, and P. M. Long. The power of localization for efficiently learning linear separators with noise. In *Proceedings of the 46th annual ACM Symposium on Theory of Computing, STOC*, pages 449–458, 2014.
- [20] J. Azimi, A. Jalali, and X. Z. Fern. Hybrid batch bayesian optimization. In *Proceedings of the 29th International Conference on Machine Learning, ICML*, volume 2, pages 1215–1222, 2012.
- [21] M.-F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In

*Proceedings of the 23rd international Conference on Machine Learning, ICML*, pages 65–72. ACM, 2006.

- [22] M.-F. Balcan, A. Broder, and T. Zhang. Margin based active learning. In *International Conference on Computational Learning Theory*, pages 35–50. Springer, 2007.
- [23] M.-F. Balcan and P. Long. Active and passive learning of linear separators under log-concave distributions. In *Proceedings of the 26th Annual Conference on Learning Theory, COLT*, pages 288–316. PMLR, 2013.
- [24] K. Ball. *An Elementary Introduction to Modern Convex Geometry*. Cambridge, 1997.
- [25] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263, 2008.
- [26] D. Basu, D. Data, C. Karakus, and S. Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification, and local computations. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, volume 32, 2019.
- [27] T. Batu and C. L. Canonne. Generalized uniformity testing. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science, FOCS*, pages 880–889. IEEE Computer Society, 2017.
- [28] T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld, and P. White. Testing random variables for independence and identity. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 442–451, 2001.
- [29] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing closeness of discrete distributions. *Journal of the ACM*, 60(1), 2013.
- [30] A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.
- [31] J. Bernstein, Y. X. Wang, K. Azizzadenesheli, and A. Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 2, pages 894–918, 2018.

- [32] A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *Proceedings of the 26th International Conference on Machine Learning, ICML*, pages 49–56, 2009.
- [33] A. Beygelzimer, D. Hsu, N. Karampatziakis, J. Langford, and T. Zhang. Efficient active learning. In *ICML Workshop on On-line Trading of Exploration and Exploitation*, 2011.
- [34] A. Beygelzimer, D. J. Hsu, J. Langford, and T. Zhang. Agnostic active learning without constraints. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems*, pages 199–207, 2010.
- [35] P. J. Bickel, Y. Ritov, and A. B. Tsybakov. Simultaneous analysis of lasso and dantzig selector. *Annals of Statistics*, 37(4):1705–1732, 2009.
- [36] I. Bistritz and A. Leshem. Game of Thrones: Fully Distributed Learning for Multi-Player Bandits. *Mathematics of Operations Research*, 46:159–178, 2021.
- [37] I. Bogunovic, S. Jegelka, J. Scarlett, and V. Cevher. Adversarially robust optimization with Gaussian processes. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems*, pages 5760–5770, 2018.
- [38] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.
- [39] S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.
- [40] D. Bouneffouf, I. Rish, and C. Aggarwal. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, page 1–8. IEEE Press, 2020.
- [41] O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In *Advanced lectures on machine learning*, pages 169–207. Springer, 2004.
- [42] J. K. Bradley, A. Kyrola, D. Bickson, and C. Guestrin. Parallel coordinate descent for L1-regularized loss minimization. In *Proceedings of the 28th International Conference on Machine Learning, ICML*, pages 321–328, 2011.
- [43] N. Bshouty. Optimal algorithms for the coin weighing problem with a

- spring scale. In *Proceedings of the 22nd Conference on Learning Theory, COLT*, 2009.
- [44] N. H. Bshouty and H. Mazzawi. Toward a deterministic polynomial time algorithm with optimal additive query complexity. *Theoretical Computer Science*, 417:23–35, 2012.
  - [45] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. X-Armed Bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.
  - [46] A. D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12:2879–2904, 2011.
  - [47] S. Cai, M. Jahangoshahi, M. Bakshi, and S. Jaggi. Grotesque: Noisy group testing (quick and efficient). In *Proceedings of the 51st Annual Allerton Conference on Communication, Control, and Computing*, pages 1234–1241, 2013.
  - [48] X. Cai and J. Scarlett. On lower bounds for standard and robust gaussian process bandit optimization. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, 2021.
  - [49] D. Calandriello, L. Carratino, A. Lazaric, M. Valko, and L. Rosasco. Gaussian Process Optimization with Adaptive Sketching: Scalable and No Regret. *Proceedings of Machine Learning Research*, 99:1–25, 2019.
  - [50] D. Calandriello, A. Lazaric, and M. Valko. Second-order kernel online convex optimization with adaptive sketching. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 2, pages 1052–1068, 2017.
  - [51] D. Calandriello and L. Rosasco. Statistical and computational trade-offs in kernel K-means. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems*, volume 2018-Decem, pages 9357–9367, 2018.
  - [52] R. Camilleri, J. Katz-Samuels, and K. Jamieson. High-Dimensional Experimental Design and Kernel Bandits. In *Proceedings of the 38 th International Conference on Machine Learning, ICML*, 2021.
  - [53] E. Candes and T. Tao. Near Optimal Signal Recovery From Random Projections : Universal Encoding Strategies. *IEEE Transactions on Information Theory*, 52:5406–5425, 2006.

- [54] E. Candes and T. Tao. The Dantzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics*, 35(6):2313–2351, 2007.
- [55] E. J. Candès, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.
- [56] C. Canonne. A Survey on Distribution Testing: Your Data is Big. But is it Blue? *Theory of Computing*, 1(1):1–100, 2020.
- [57] A. Carpentier and R. Munos. Bandit Theory meets Compressed Sensing for high-dimensional Stochastic Linear Bandit. *Journal of Machine Learning Research*, 22:190–198, 2012.
- [58] R. M. Castro and R. D. Nowak. Minimax bounds for active learning. *IEEE Transactions on Information Theory*, 54(5):2339–2353, 2008.
- [59] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear classification and selective sampling under low noise conditions. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, pages 249–256, 2009.
- [60] S. Cen, H. Zhang, Y. Chi, W. Chen, and T. Y. Liu. Convergence of Distributed Stochastic Variance Reduced Methods without Sampling Extra Data. *IEEE Transactions on Signal Processing*, 68:3976–3989, 2020.
- [61] N. Cesa-Bianchi, A. Conconi, and C. Gentile. Learning probabilistic linear-threshold classifiers via selective sampling. In *Proceedings of the 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel*, pages 373–387. Springer, 2003.
- [62] C. L. Chan, S. Jaggi, V. Saligrama, and S. Agnihotri. Non-adaptive group testing: Explicit bounds and novel algorithms. *IEEE Transactions on Information Theory*, 60(5):3019–3035, 2014.
- [63] S. O. Chan, I. Diakonikolas, P. Valiant, and G. Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1193–1203, 2014.
- [64] R. Chawla, A. Sankararaman, A. Ganesh, and S. Shakkottai. The Gossiping Insert-Eliminate Algorithm for Multi-Agent Bandits. In *Proceedings of*

*the 23rd International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 3471–3481. PMLR, 2020.

- [65] R. Chawla, A. Sankararaman, and S. Shakkottai. Multi-Agent Low-Dimensional Linear Bandits. *IEEE Transactions on Automatic Control*, 2022.
- [66] C. Chen, L. Luo, W. Zhang, Y. Yu, and Y. Lian. Efficient and robust high-dimensional linear contextual bandits. In *Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence*, pages 4259–4265, 2021.
- [67] Y. Chen, Y. Wang, E. X. Fang, Z. Wang, and R. Li. Nearly Dimension-Independent Sparse Linear Bandit over Small Action Spaces via Best Subset Selection. *Journal of the American Statistical Association*, pages 1–31, 2022.
- [68] S. R. Chowdhury and A. Gopalan. On kernelized multi-armed bandits. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 2, pages 1397–1422, 2017.
- [69] W. Chu, L. Li, L. Reyzin, and R. Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.
- [70] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.
- [71] E. Contal, D. Buffoni, A. Robicquet, and N. Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 225–240, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [72] E. Contal and N. Vayatis. Stochastic Process Bandits: Upper Confidence Bounds Algorithms via Generic Chaining, 2016.
- [73] C. Cortes, G. DeSalvo, C. Gentile, M. Mohri, and N. Zhang. Region-based active learning. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 2801–2809, 2019.
- [74] T. M. Cover and J. A. Thomas. *Elements of Information Theory* (Wiley Se-

- ries in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006.
- [75] H. Cramér. Sur un nouveau théorème-limite de la théorie des probabilités. *Actual. sci. industr.* 736, 5-23. (Confér. internat. Sci. math. Univ. Genève. Théorie des probabilités. III: Les sommes et les fonctions de variables aléatoires.), 1938.
  - [76] D. Csiba, Z. Qu, and P. Richtarik. Stochastic dual coordinate ascent with adaptive probabilities. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, volume 1, pages 674–683, 2015.
  - [77] X. Dai, X. Yan, K. Zhou, H. Yang, K. K. W. Ng, J. Cheng, and Y. Fan. Hyper-Sphere Quantization: Communication-Efficient SGD for Federated Learning, 2019.
  - [78] Z. Dai, B. K. H. Low, and P. Jaillet. Federated Bayesian optimization via Thompson sampling. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, volume 2020-Decem, 2020.
  - [79] C. D. Dang and G. Lan. Stochastic block mirror descent methods for nonsmooth and stochastic optimization. *SIAM Journal on Optimization*, 25(2):856–881, 2015.
  - [80] V. Dani, T. P. Hayes, and S. M. Kakade. The price of bandit information for online optimization. In *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems*, 2008.
  - [81] A. Daniely, R. Frostig, and Y. Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems*, pages 2261–2269, 2016.
  - [82] R. Das, A. Acharya, A. Hashemi, S. Sanghavi, I. S. Dhillon, and U. Topcu. Faster Non-Convex Federated Learning via Global and Local Momentum. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence, UAI*, pages 496–506. Association For Uncertainty in Artificial Intelligence (AUAI), 2022.
  - [83] S. Dasgupta. Two faces of active learning. *Theoretical computer science*, 412(19):1767–1781, 2011.

- [84] S. Dasgupta, D. J. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems*, pages 353–360, 2008.
- [85] S. Dasgupta, A. T. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In *International Conference on Computational Learning Theory*, pages 249–263. Springer, 2005.
- [86] F. N. David. Two combinatorial tests of whether a sample has come from a given population. *Biometrika*, 37(1-2):97–110, 1950.
- [87] N. de Bruijn. *Asymptotic Methods in Analysis*. Bibliotheca mathematica. Dover Publications, 1981.
- [88] N. De Freitas, A. J. Smola, and M. Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In *Proceedings of the 29th International Conference on Machine Learning, ICML*, volume 2, pages 1743–1750, 2012.
- [89] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202, 2012.
- [90] Q. Deng, J. Ho, and A. Rangarajan. Stochastic coordinate descent for non-smooth convex optimization. *27th Annual Conference on Neural Information Processing Systems Workshop on Optimization*, 2013.
- [91] Y. Deng, M. M. Kamani, and M. Mahdavi. Adaptive Personalized Federated Learning, 2020.
- [92] T. Desautels, A. Krause, and J. Burdick. Parallelizing exploration-exploitation tradeoffs with Gaussian process bandit optimization. In *Proceedings of the 29th International Conference on Machine Learning, ICML*, volume 2, pages 1191–1198, 2012.
- [93] I. Diakonikolas, T. Gouleakis, J. Peebles, and E. Price. Sample-optimal identity testing with high probability. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 107, 2018.
- [94] I. Diakonikolas, E. Grigorescu, J. Li, A. Natarajan, K. Onak, and L. Schmidt. Communication-efficient distributed learning of discrete dis-

- tributions. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, volume 30, 2017.
- [95] I. Diakonikolas, D. M. Kane, and V. Nikishkin. Testing Identity of Structured Distributions. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1841–1854. SIAM, 2015.
  - [96] C. T. Dinh, N. H. Tran, T. D. Nguyen, W. Bao, A. Y. Zomaya, and B. B. Zhou. Federated Learning with Proximal Stochastic Variance Reduced Gradient Algorithms. In *ACM International Conference Proceeding Series*, volume 20. Association for Computing Machinery, 2020.
  - [97] A. Djakov. On a search model of false coins. *Topics in Information Theory (Colloquia Mathematica Societatis Janos Bolyai)*, 16:163–170, 1975.
  - [98] R. Dorfman. The Detection of Defective Members of Large Populations. *The Annals of Mathematical Statistics*, 14(4):436–440, 1943.
  - [99] Y. Du, W. Chen, Y. Kuroki, and L. Huang. Collaborative Pure Exploration in Kernel Bandit. In *Proceedings of the 11th International Conference on Learning Representations, ICLR*, 2023.
  - [100] D. Dua and C. Graff. UCI machine learning repository, 2017.
  - [101] A. Dubey and A. Pentland. Differentially-private federated linear bandits. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, volume 2020-Decem, 2020.
  - [102] A. Dubey and A. Pentland. Kernel methods for cooperative multi-agent contextual bandits. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, pages 2720–2730, 2020.
  - [103] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and Y. Zhang. Optimality guarantees for distributed statistical estimation, 2014.
  - [104] J. N. Eberhardt, N. P. Breuckmann, and C. S. Eberhardt. Multi-Stage Group Testing Improves Efficiency of Large-Scale COVID-19 Screening. *Journal of Clinical Virology*, 128, 2020.
  - [105] P. Erdős and A. Rényi. On two problems in information theory. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 8:229–243, 1963.

- [106] A. Fallah, A. Mokhtari, and A. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, volume 33, 2020.
- [107] O. Fawzi, N. Flammarion, A. Garivier, and A. Oufkir. Sequential algorithms for testing identity and closeness of distributions. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems*, 2022.
- [108] O. Fercoq and P. Richtárik. Smooth Minimization of Nonsmooth Functions with Parallel Coordinate Descent Methods. In *Springer Proceedings in Mathematics and Statistics*, volume 279, pages 57–96, 2019.
- [109] M. C. Ferris and O. L. Mangasarian. Parallel Variable Distribution. *SIAM Journal on Optimization*, 4(4):815–832, 1994.
- [110] N. Fine. Solution of problem e 1399. *The American Mathematical Monthly*, 67(7):697–698, 1960.
- [111] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2-3):133–168, 1997.
- [112] N. Gantert, K. Ramanan, and F. Rembart. Large deviations for weighted sums of stretched exponential random variables. *Electronic Communications in Probability*, 19, 2014.
- [113] L. Gargano, V. Montuori, G. Setaro, and U. Vaccaro. An improved algorithm for quantitative group testing. *Discrete Applied Mathematics*, 36(3):299–306, 1992.
- [114] O. Gebhard, M. Hahn-Klimroth, D. Kaaser, and P. Loick. Quantitative group testing in the sublinear regime, 2019.
- [115] A. Ghosh, A. Sankararaman, and K. Ramchandran. Adaptive Clustering and Personalization in Multi-Agent Stochastic Linear Bandits, 2021.
- [116] O. Goldreich and D. Ron. *On testing expansion in bounded-degree graphs*, pages 68–75. Springer Berlin, 2011.
- [117] S. Greenberg and M. Mohri. Tight lower bound on the probability of a

- binomial exceeding its expectation. *Statistics & Probability Letters*, 86:91 – 98, 2014.
- [118] L. Grippo and M. Sciandrone. Globally convergent block-coordinate techniques for unconstrained optimization. *Optimization Methods and Software*, 10(4):587–637, 1999.
  - [119] S. Grünewälder, J. Y. Audibert, M. Opper, and J. Shawe-Taylor. Regret bounds for Gaussian process bandit problems. *Journal of Machine Learning Research*, 9:273–280, 2010.
  - [120] Q. Gu, A. Karbasi, K. Khosravi, V. Mirrokni, and D. Zhou. Batched neural bandits, 2021.
  - [121] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. R. Cadambe. Local SGD with periodic averaging: Tighter analysis and adaptive synchronization. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, volume 32, 2019.
  - [122] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi. Federated learning with compression: Unified analysis and sharp guarantees. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 2350–2358. PMLR, 2021.
  - [123] O. A. Hanna, L. F. Yang, and C. Fragouli. Solving Multi-Arm Bandit Using a Few Bits of Communication. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 11215–11236. PMLR, 2022.
  - [124] J. Hannan. Approximation to rayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.
  - [125] S. Hanneke. A bound on the label complexity of agnostic active learning. In *Proceedings of the 24th International Conference on Machine Learning, ICML*, pages 353–360, 2007.
  - [126] S. Hanneke. Adaptive rates of convergence in active learning. In *Proceedings of the 22nd Conference on Learning Theory, COLT*. Citeseer, 2009.
  - [127] S. Hanneke et al. Rates of convergence in active learning. *The Annals of Statistics*, 39(1):333–361, 2011.

- [128] S. Hanneke et al. Theory of disagreement-based active learning. *Foundations and Trends® in Machine Learning*, 7(2-3):131–309, 2014.
- [129] S. Hanneke and L. Yang. Surrogate losses in passive and active learning, 2012.
- [130] S. Hanneke and L. Yang. Minimax analysis of active learning. *Journal of Machine Learning Research*, 16(1):3487–3602, 2015.
- [131] F. Hanzely and P. Richtárik. Federated Learning of a Mixture of Global and Local Models, 2020.
- [132] B. Hao, T. Lattimore, and M. Wang. High-dimensional sparse linear bandits. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, volume 2020-Decem, 2020.
- [133] F. H. Hao. The optimal procedures for quantitative group testing. *Discrete Applied Mathematics*, 26(1):79–86, 1990.
- [134] J. Haupt, R. M. Castro, and R. Nowak. Distilled sensing: Adaptive sampling for sparse detection and estimation. *IEEE Transactions on Information Theory*, 57(9):6222–6235, 2011.
- [135] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, UAI*, pages 282–290, 2013.
- [136] E. Hillel, Z. Karnin, T. Koren, R. Lempel, and O. Somekh. Distributed exploration in Multi-Armed Bandits. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, 2013.
- [137] M. D. Hoffman, E. Brochu, and N. de Freitas. Portfolio allocation for bayesian optimization. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, UAI*, pages 327–336, 2011.
- [138] R. Höning, Y. Zhao, and R. Mullins. DAdaQuant: Doubly-adaptive quantization for communication-efficient federated learning. In *Proceedings of the 39th International Conference on Machine Learning, ICML*, volume 162, pages 8852–8866. PMLR, 2022.
- [139] C. J. Hsieh, K. W. Chang, C. J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceed-*

*ings of the 25th International Conference on Machine Learning*, pages 408–415, 2008.

- [140] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines. *Journal of Machine Learning Research*, 9:1369–1398, 2008.
- [141] B. Huang, S. Salgia, and Q. Zhao. Disagreement-based active learning in online settings. *IEEE Transactions on Signal Processing*, 70:1947–1958, 2022.
- [142] D. Huang and S. Meyn. Generalized Error Exponents For Small Sample Universal Hypothesis Testing. *IEEE Transactions on Information Theory*, 59(12):8157–8181, 2013.
- [143] R. Huang, W. Wu, J. Yang, and C. Shen. Federated Linear Contextual Bandits. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems*, volume 32, pages 27057–27068, 2021.
- [144] J. H. Huggins, T. Campbell, M. Kasprzak, and T. Broderick. Scalable Gaussian process inference with finite-data mean and variance guarantees. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, AISTATS*, 2019.
- [145] F. K. Hwang. A tale of two coins. *The American Mathematical Monthly*, 94(2):121–129, 1987.
- [146] Y. I. Ingster. Adaptive Chi-Square Tests. *Zapiski Nauchnykh Seminarov POMI*, 244(2):150–166, 1997.
- [147] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems*, pages 8571–8580, 2018.
- [148] D. Janz, D. R. Burt, and J. González. Bandit optimisation of functions in the Matérn kernel RKHS, 2020.
- [149] D. Jhunjhunwala, A. Gadhikar, G. Joshi, and Y. C. Eldar. Adaptive quantization of model updates for communication-efficient federated learning. In *Proceedings of the 46th IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 3110–3114. IEEE, 2021.

- [150] Y. Jiang, J. Konečný, K. Rush, and S. Kannan. Improving Federated Learning Personalization via Model Agnostic Meta Learning, 2019.
- [151] C. Jin, P. Netrapalli, R. Ge, S. M. Kakade, and M. I. Jordan. A short note on concentration inequalities for random vectors with subgaussian norm, 2019.
- [152] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. D’Oliveira, H. Eichner, S. El Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecní, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning, 2021.
- [153] D. Kalathil, N. Nayyar, and R. Jain. Decentralized learning for multi-player multi-armed bandits. In *Proceedings of the 51st IEEE Conference on Decision and Control*, pages 3960–3965, 2012.
- [154] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur. Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences, 2018.
- [155] K. Kandasamy, G. Dasarathy, J. Oliva, J. Schneider, and B. Póczos. Multi-fidelity Gaussian process bandit optimisation. *Journal of Artificial Intelligence Research*, 66:151–196, 2019.
- [156] K. Kandasamy, A. Krishnamurthy, J. Schneider, and B. Póczos. Parallelised Bayesian optimisation via Thompson sampling. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, pages 133–142, 2018.
- [157] K. Kandasamy, J. Schneider, and B. Póczos. High dimensional Bayesian Optimisation and bandits via additive models. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, volume 1, pages 295–304, 2015.
- [158] E. Karimi, F. Kazemi, A. Heidarzadeh, K. R. Narayanan, and A. Sprintson. Non-adaptive quantitative group testing using irregular sparse graph

- codes. In *Proceedings of the 57th Annual Allerton Conference on Communication, Control, and Computing*, pages 608–614, 2019.
- [159] H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9851 LNAI, pages 795–811, 2016.
- [160] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. Mime: Mimicking Centralized Stochastic Algorithms in Federated Learning, 2020.
- [161] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume PartF16814, pages 5088–5099, 2020.
- [162] P. Kassraie and A. Krause. Neural contextual bandits without regret. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2022.
- [163] K. Kawaguchi, L. P. Kaelbling, and T. Lozano-Pérez. Bayesian optimization with exponential convergence. In *Proceedings of the 29th Conference on Advances in Neural Information Processing Systems*, volume 2015-Janua, pages 2809–2817, 2015.
- [164] A. Khaled, K. Mishchenko, and P. Richtárik. Tighter Theory for Local SGD on Identical and Heterogeneous Data. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 4519–4529. PMLR, 2020.
- [165] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466, 1952.
- [166] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 681–690, 2008.
- [167] V. Koltchinskii. Rademacher complexities and bounding the excess risk in active learning. *Journal of Machine Learning Research*, 11(Sep):2457–2485, 2010.

- [168] V. Koltchinskii et al. Local rademacher complexities and oracle inequalities in risk minimization. *The Annals of Statistics*, 34(6):2593–2656, 2006.
- [169] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated Learning: Strategies for Improving Communication Efficiency, 2016.
- [170] J. Konečný, J. Liu, P. Richtárik, and M. Takáč. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255, 2016.
- [171] N. Korda, B. Szorenyi, and S. Li. Distributed clustering of linear bandits in peer to peer networks. In *Proceedings of the 33rd International Conference on Machine Learning, ICML*, volume 3, pages 1966–1980, 2016.
- [172] A. Krause and C. S. Ong. Contextual Gaussian process bandit optimization. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems, NIPS*, pages 2447–2455, 2011.
- [173] V. Kulkarni, M. Kulkarni, and A. Pant. Survey of personalization techniques for federated learning. In *Proceedings of the World Conference on Smart Trends in Systems, Security and Sustainability, WS4 2020*, pages 794–797, 2020.
- [174] I. Kuzborskij, L. Cella, and N. Cesa-Bianchi. Efficient linear bandits through matrix sketching. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, AISTATS*, 2019.
- [175] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.
- [176] P. Landgren, V. Srivastava, and N. E. Leonard. On distributed cooperative decision-making in multiarmed bandits. In *Proceedings of the 5th European Control Conference, ECC*, pages 243–248. Institute of Electrical and Electronics Engineers Inc., 2017.
- [177] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*, volume 20, 2007.
- [178] T. Lattimore and C. Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

- [179] L. LeCam. Convergence of Estimates Under Dimensionality Restrictions. *The Annals of Statistics*, 1(1):38–53, 1973.
- [180] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [181] J. D. Lee, Q. Lin, T. Ma, and T. Yang. Distributed stochastic variance reduced gradient methods by sampling extra data with replacement. *Journal of Machine Learning Research*, 18(122):1–43, 2017.
- [182] K. Lee, R. Pedarsani, and K. Ramchandran. Saffron: A fast, efficient, and robust framework for group testing based on sparse-graph codes. In *Proceedings of the 49th IEEE International Symposium on Information Theory, ISIT*, pages 2873–2877, 2016.
- [183] M. Lee, S. Shekhar, and T. Javidi. Multi-Scale Zero-Order Optimization of Smooth Functions in an RKHS. In *IEEE International Symposium on Information Theory - Proceedings*, volume 2022-June, pages 288–293, 2022.
- [184] D. Leventhal and A. S. Lewis. Randomized methods for linear constraints: Convergence rates and conditioning. *Mathematics of Operations Research*, 35(3):641–654, 2010.
- [185] B. Li, S. Tang, and H. Yu. Better approximations of high dimensional smooth functions by deep neural networks with rectified power units. *Communications in Computational Physics*, 27(2):379–411, 2019.
- [186] C. Li, G. Li, and P. K. Varshney. Communication-efficient federated learning based on compressed sensing. *IEEE Internet of Things Journal*, 8(20):15531–15541, 2021.
- [187] C. Li, H. Wang, M. Wang, and H. Wang. Communication Efficient Distributed Learning for Kernelized Contextual Bandits. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems*, 2022.
- [188] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web, WWW*, pages 661–670, 2010.
- [189] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Chal-

- lenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.
- [190] T. Li, L. Song, and C. Fragouli. Federated Recommendation System via Differential Privacy. In *Proceedings of the 53rd IEEE International Symposium on Information Theory, ISIT*, volume 2020-June, pages 2592–2597, 2020.
- [191] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang. On the Convergence of FedAvg on Non-IID Data. In *Proceedings of the 8th International Conference on Learning Representations, ICLR*, 2020.
- [192] Y. Li, Y. Wang, and Y. Zhou. Nearly minimax-optimal regret for linearly parameterized bandits. In *Conference on Learning Theory*, pages 2173–2174. PMLR, 2019.
- [193] Z. Li and J. Scarlett. Gaussian process bandit optimization with few batches. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2022.
- [194] Z. Li, H. Zhao, B. Li, and Y. Chi. SoteriaFL: A Unified Framework for Private Federated Learning with Communication Compression. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems*, 2022.
- [195] J. Lipor, B. P. Wong, D. Scavia, B. Kerkez, and L. Balzano. Distance-penalized active learning using quantile search. *IEEE Transactions on Signal Processing*, 65(20):5453–5465, 2017.
- [196] C. Liu, L. Zhu, and M. Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, volume 33, pages 15954–15964, 2020.
- [197] J. Liu, S. J. Wright, C. Ré, and S. Sridhar. An Asynchronous Parallel Stochastic Coordinate Descent Algorithm. *Journal of Machine Learning Research*, 16:285–322, 2015.
- [198] K. Liu and Q. Zhao. Distributed learning in multi-armed bandit with multiple players. *IEEE Transactions on Signal Processing*, 58(11):5667–5681, 2010.
- [199] Y. Liu, X. Zhang, Y. Kang, L. Li, T. Chen, M. Hong, and Q. Yang. Fedbcd: A

- communication-efficient collaborative learning framework for distributed features. *IEEE Transactions on Signal Processing*, 70:4277–4290, 2022.
- [200] Z. Lu and L. Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, 152(1-2):615–642, 2015.
  - [201] Z. Q. Luo and P. Tseng. On the Convergence of the Coordinate Descent Method for Convex Differentiable Minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.
  - [202] A. Mahmoudi, J. M. B. D. S. Junior, H. S. Ghadikolaei, and C. Fischione. A-LAQ: Adaptive Lazily Aggregated Quantized Gradient. In *Proceedings of the 18th IEEE GLOBECOM Workshops*, pages 1828–1833, 2022.
  - [203] D. Malioutov and M. Malyutov. Boolean compressed sensing:  $L_p$  relaxation for group testing. In *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 3305–3308, 2012.
  - [204] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh. Three Approaches for Personalization with Applications to Federated Learning, 2020.
  - [205] J. Mareček, P. Richtárik, and M. Takáč. Distributed block coordinate descent for minimizing partially separable functions. In *Springer Proceedings in Mathematics and Statistics*, volume 134, pages 261–288, 2015.
  - [206] P. Massart, É. Nédélec, et al. Risk bounds for statistical learning. *The Annals of Statistics*, 34(5):2326–2366, 2006.
  - [207] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 1273–1282. PMLR, 2017.
  - [208] B. McWilliams, D. Balduzzi, and J. M. Buhmann. Correlated random features for fast semi-supervised learning. In *Proceedings of the 27th Conference on Advances in Neural Information Processing Systems*, 2013.
  - [209] K. Mishchenko, G. Malinovsky, S. Stich, and P. Richtárik. ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!, 2022.

- [210] A. Mitra, A. Adibi, G. J. Pappas, and H. Hassani. Collaborative Linear Bandits with Adversarial Agents: Near-Optimal Regret Bounds. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems*, 2022.
- [211] A. Mitra, H. Hassani, and G. Pappas. Exploiting Heterogeneity in Robust Federated Best-Arm Identification, 2021.
- [212] A. Mitra, H. Hassani, and G. J. Pappas. Linear Stochastic Bandits over a Bit-Constrained Channel, 2022.
- [213] R. Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, 2011.
- [214] M. Mutný and A. Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature fourier features. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems*, volume 2018-Decem, pages 9005–9016, 2018.
- [215] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.
- [216] Y. Nesterov. Subgradient methods for huge-scale optimization problems. *Mathematical Programming*, 146(1-2):275–297, 2014.
- [217] V. Nguyen, S. Gupta, S. Rana, C. Li, and S. Venkatesh. Regret for expected improvement over the best-observed value and stopping condition. In *Proceedings of the Asian Conference on Machine Learning*, pages 279–294. PMLR, 2017.
- [218] J. Opschoor, C. Schwab, and J. Zech. Exponential relu dnn expression of holomorphic maps in high dimension. *Constructive Approximation*, 55:1–46, 2022.
- [219] J. Paisley, X. Liao, and L. Carin. Active learning and basis selection for kernel-based linear models: A bayesian perspective. *IEEE Transactions on Signal Processing*, 58(5):2686–2700, 2010.
- [220] L. Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Transactions on Information Theory*, 54(10):4750–4755, 2008.

- [221] R. Pasupathy, V. Tech, and S. Kim. The Stochastic Root Finding Problem: Overview, Solutions, and Open Questions. *ACM Transactions on Modeling and Computational Simulations*, 21(3):19, 2011.
- [222] R. Pathak and M. J. Wainwright. FedSplit: An algorithmic framework for fast federated optimization. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, volume 2020-Decem, 2020.
- [223] Z. Peng, M. Yan, and W. Yin. Parallel and distributed sparse optimization. In *Proceedings of the 13th Asilomar Conference on Signals, Systems and Computers*, pages 659–664, 2013.
- [224] V. Picheny, S. Vakili, and A. Artemev. Ordinal bayesian optimisation, 2019.
- [225] V. Picheny, T. Wagner, and D. Ginsbourger. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization*, 48(3):607–626, 2013.
- [226] S. Pouriyeh, O. Shahid, R. M. Parizi, Q. Z. Sheng, G. Srivastava, L. Zhao, and M. Nasajpour. Secure Smart Communication Efficiency in Federated Learning: Achievements and Challenges. *Applied Sciences (Switzerland)*, 12(18):10996, 2022.
- [227] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, 2009.
- [228] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.
- [229] M. Razaviyayn, M. Hong, and Z. Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.
- [230] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan. Adaptive Federated Optimization. In *Proceedings of the 9th International Conference on Learning Representations, ICLR*, 2021.
- [231] S. J. Reddi, A. Hefny, C. Downey, A. Dubey, and S. Sra. Large-scale randomized-coordinate descent methods with non-separable linear con-

- straints. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence, UAI*, pages 762–771, 2015.
- [232] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 2021–2031. PMLR, 2020.
  - [233] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.
  - [234] P. Richtárik and M. Takáč. Distributed coordinate descent method for learning with big data. *Journal of Machine Learning Research*, 17:1–25, 2016.
  - [235] P. Richtárik and M. Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6):1233–1243, 2016.
  - [236] P. Richtárik and M. Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484, 2016.
  - [237] P. Rigollet and J.-C. Hütter. High Dimensional Statistics Lecture Notes, 2017.
  - [238] C. Riquelme, G. Tucker, and J. Snoek. Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for Thompson sampling. In *Proceedings of the 6th International Conference on Learning Representations, ICLR*, 2018.
  - [239] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Statistics*, 22(3):400–407, 1951.
  - [240] J. Rosenski, O. Shamir, and L. Szlak. Multi-player bandits - A musical chairs approach. In *Proceedings of the 33rd International Conference on Machine Learning, ICML*, volume 1, pages 276–298, 2016.
  - [241] S. Ruder. An overview of gradient descent optimization algorithms, 2016.
  - [242] P. Rusmevichientong and J. N. Tsitsiklis. Linearly parameterized bandits. *Mathematics of Operations Research*, 35(2):395–411, 2010.

- [243] D. Russo and B. Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.
- [244] D. Russo and B. Van Roy. An information-theoretic analysis of Thompson sampling. *The Journal of Machine Learning Research*, 17(1), 2016.
- [245] A. Saha and A. Tewari. On the finite time convergence of cyclic coordinate descent methods. *SIAM Journal on Optimization*, 23(1):576–601, 2013.
- [246] F. Salehi, P. Thiran, and L. Elisa Celis. Coordinate descent with bandit sampling. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems*, pages 9247–9257, 2018.
- [247] S. Salgia, S. Vakili, and Q. Zhao. A domain-shrinking based Bayesian optimization algorithm with order-optimal regret performance. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems*, volume 34, 2021.
- [248] S. Salgia, S. Vakili, and Q. Zhao. Collaborative Learning in Kernel-based Bandits for Distributed Users, 2022.
- [249] S. Salgia, S. Vakili, and Q. Zhao. Provably and practically efficient neural contextual bandits, 2022.
- [250] S. Salgia and Q. Zhao. An order-optimal adaptive test plan for noisy group testing under unknown noise models. In *Proceedings of the 46th International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 4035–4039, 2021.
- [251] S. Salgia and Q. Zhao. Distributed linear bandits under communication constraints, 2022.
- [252] S. Salgia, Q. Zhao, and L. Tong. As Easy as ABC: Adaptive Binning Coincidence Test for Uniformity Testing, 2021.
- [253] S. Salgia, Q. Zhao, and S. Vakili. Stochastic coordinate minimization with progressive precision for stochastic convex optimization. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume PartF16814, pages 8396–8406, 2020.
- [254] A. Sankararaman, A. Ganesh, and S. Shakkottai. Social learning in multi

- agent multi armed bandits. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3), 2019.
- [255] P. Sattari, M. Kurant, A. Anandkumar, A. Markopoulou, and M. G. Rabbat. Active learning of multiple source multiple destination topologies. *IEEE Transactions on Signal Processing*, 62(8):1926–1937, 2014.
  - [256] J. Scarlett. Tight regret bounds for Bayesian optimization in one dimension. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, 2018.
  - [257] J. Scarlett, I. Bogunovic, and V. Cevher. Lower Bounds on Regret for Noisy Gaussian Process Bandit Optimization. In *Conference on Learning Theory*, volume 65, pages 1–20, 2017.
  - [258] J. Scarlett and V. Cevher. Phase transitions in group testing. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, page 40–53, USA, 2016. Society for Industrial and Applied Mathematics.
  - [259] J. Scarlett and V. Cevher. Near-optimal noisy group testing via separate decoding of items. In *Proceedings of the 51st IEEE International Symposium on Information Theory (ISIT)*, pages 2311–2315, 2018.
  - [260] M. Seeger, C. Williams, and N. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Proceedings of the 6th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2003.
  - [261] B. Settles. *Active learning*. Springer Cham, 2012.
  - [262] S. Shahrampour, A. Rakhlin, and A. Jadbabaie. Multi-armed bandits in multi-agent networks. In *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 2786–2790, 2017.
  - [263] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.
  - [264] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.
  - [265] S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual

- coordinate ascent for regularized loss minimization. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning, ICML*, volume 32 of *Proceedings of Machine Learning Research*, pages 64–72, Beijing, China, 2014. PMLR.
- [266] H. S. Shapiro. Problem e 1399. *The American Mathematical Monthly*, 67(82):697–697, 1960.
  - [267] S. Shekhar and T. Javidi. Gaussian process bandits with adaptive discretization. *Electronic Journal of Statistics*, 12(2):3829–3874, 2018.
  - [268] S. Shekhar and T. Javidi. Significance of gradient information in bayesian optimization. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, pages 2836–2844. PMLR, 2021.
  - [269] C. Shi and C. Shen. Federated Multi-Armed Bandits. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pages 9603–9611, 2021.
  - [270] C. Shi, C. Shen, and J. Yang. Federated Multi-armed Bandits with Personalization. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 2917–2925. PMLR, 2021.
  - [271] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, volume 33, pages 7462–7473, 2020.
  - [272] V. Smith, C. K. Chiang, M. Sanjabi, and A. Talwalkar. Federated multi-task learning. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, pages 4425–4435, 2017.
  - [273] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, volume 25, 2012.
  - [274] A. Sobester, A. Forrester, and A. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, 2008.
  - [275] S. Soderberg and H. S. Shapiro. A combinatory detection problem. *The American Mathematical Monthly*, 70(10):1066–1070, 1963.

- [276] F. Sossan, E. Namor, R. Cherkaoui, and M. Paolone. Achieving the dispatchability of distribution feeders through prosumers data driven forecasting and model predictive control of electrochemical storage. *IEEE Transactions on Sustainable Energy*, 7(4):1762–1777, 2016.
- [277] A. Spiridonoff, A. Olshevsky, and I. C. Paschalidis. Local SGD With a Communication Overhead Depending Only on the Number of Workers, 2020.
- [278] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning, ICML*, pages 1015–1022, 2010.
- [279] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.
- [280] S. Stich. Local SGD converges fast and communicates little. In *Proceedings of the 7th International Conference on Learning Representations, ICLR*, 2019.
- [281] J. Sun, T. Chen, G. Giannakis, and Z. Yang. Communication-efficient distributed learning via lazily aggregated quantized gradients. *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 32, 2019.
- [282] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan. Distributed mean estimation with limited communication. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 7, pages 5119–5128, 2017.
- [283] H. Tang, X. Lian, C. Yu, T. Zhang, and J. Liu. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pages 10747–10757, 2019.
- [284] Z. Tang, S. Shi, X. Chu, W. Wang, and B. Li. Communication-efficient distributed deep learning: A comprehensive survey, 2020.
- [285] C. Tao, Q. Zhang, and Y. Zhou. Collaborative learning with limited interaction: Tight bounds for distributed exploration in multi-Armed bandits. In *Proceedings of the 60th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, volume 2019-Novem, pages 126–146, 2019.

- [286] Q. Tao, K. Kong, D. Chu, and G. Wu. Stochastic coordinate descent methods for regularized smooth and nonsmooth losses. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7523, pages 537–552, 2012.
- [287] R. Tappenden, P. Richtárik, and J. Gondzio. Inexact Coordinate Descent: Complexity and Preconditioning. *Journal of Optimization Theory and Applications*, 170(1):144–176, 2016.
- [288] A. Tewari and S. Shalev-Shwartz. Stochastic Methods for  $\ell_1$ -regularized Loss Minimization. *Journal of Machine Learning Research*, 12:1865–1892, 2011.
- [289] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1):267–288, 1996.
- [290] R. Tibshirani. Coordinate descent, 2013.
- [291] M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. *Journal of Machine Learning Research*, 5:567–574, 2009.
- [292] A. Tsakmalis, S. Chatzinotas, and B. Ottersten. Constrained bayesian active learning of a linear classifier. In *Proceedings of the 43rd International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 6663–6667, 2018.
- [293] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.
- [294] P. Tseng and S. Yun. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of Optimization Theory and Applications*, 140(3):513, 2008.
- [295] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.
- [296] T. Tsiligkaridis, B. M. Sadler, and A. O. Hero. Collaborative 20 questions for target localization. *IEEE Transactions on Information Theory*, 60(4):2233–2252, 2014.

- [297] J. N. Tsitsiklis and Z. Q. Luo. Communication complexity of convex optimization. *Journal of Complexity*, 3(3):231–243, 1987.
- [298] A. B. Tsybakov et al. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32(1):135–166, 2004.
- [299] S. Vakili, N. Bouziani, S. Jalali, A. Bernacchia, and D.-s. Shiu. Optimal order simple regret for Gaussian process bandits. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems*, 2021.
- [300] S. Vakili, M. Bromberg, J. Garcia, D.-s. Shiu, and A. Bernacchia. Uniform generalization bounds for overparameterized neural networks, 2021.
- [301] S. Vakili, K. Khezeli, and V. Picheny. On information gain and regret bounds in Gaussian process bandits. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2021.
- [302] S. Vakili, K. Liu, and Q. Zhao. Deterministic sequencing of exploration and exploitation for multi-armed bandit problems. *IEEE Journal on Selected Topics in Signal Processing*, 7(5):759–767, 2013.
- [303] S. Vakili, H. Moss, A. Artemev, V. Dutordoir, and V. Picheny. Scalable Thompson Sampling using Sparse Gaussian Process Models. In *Proceedings of the 35th Conference on Advances in Neural Information Processing Systems*, volume 7, pages 5631–5643, 2021.
- [304] S. Vakili, S. Salgia, and Q. Zhao. Stochastic Gradient Descent on a Tree: An Adaptive and Robust Approach to Stochastic Convex Optimization. In *Proceedings of the 57th Annual Allerton Conference on Communication, Control, and Computing, Allerton*, pages 432–438, 2019.
- [305] S. Vakili, J. Scarlett, and T. Javidi. Open problem: Tight online confidence intervals for RKHS elements. In *Conference on Learning Theory*, pages 4647–4652. PMLR, 2021.
- [306] S. Vakili and Q. Zhao. A random walk approach to first-order stochastic convex optimization. In *Proceedings of the 62nd IEEE International Symposium on Information Theory, ISIT*, pages 395–399, 2019.
- [307] S. Vakili, Q. Zhao, C. Liu, and C. Chuah. Hierarchical heavy hitter detection under unknown models. In *Proceedings of the 43rd IEEE International*

*Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6917–6921, 2018.

- [308] M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini. Finite-time analysis of kernelised contextual bandits. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, UAI*, pages 654–663, 2013.
- [309] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.
- [310] I. I. Viktorova and V. P. Chistyakov. Some Generalizations of the Test of Empty Boxes. *Theory of Probability & Its Applications*, 11(2):270–276, 1966.
- [311] C. Wang, K. Cohen, and Q. Zhao. Active hypothesis testing on a tree: Anomaly detection under hierarchical observations. In *Proceedings of the 50th IEEE International Symposium on Information Theory (ISIT)*, pages 993–997, 2017.
- [312] C. Wang, Q. Zhao, and C. N. Chuah. Optimal Nested Test Plan for Combinatorial Quantitative Group Testing. *IEEE Transactions on Signal Processing*, 66(4):992–1006, 2018.
- [313] C. Wang, Q. Zhao, and K. Cohen. Dynamic Search on a Tree with Information-Directed Random Walk. In *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, volume 2018-June, pages 1–5. IEEE, 2018.
- [314] H. Wang and A. Banerjee. Randomized Block Coordinate Descent for Online and Stochastic Optimization, 2014.
- [315] J. Wang and G. Joshi. Cooperative SGD: A Unified Framework for the Design and Analysis of Local-Update SGD Algorithms. *Journal of Machine Learning Research*, 22(213):1–50, 2021.
- [316] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage. Federated Evaluation of On-device Personalization, 2019.
- [317] L. Wang, R. Fonseca, and Y. Tian. Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search. In *Proceedings of the 34th Conference on Advances in Neural Information Processing Systems*. arXiv, 2020.

- [318] T. Wang, D. Zhou, and Q. Gu. Provably Efficient Reinforcement Learning with Linear Function Approximation under Adaptivity Constraints. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems*, volume 17, pages 13524–13536, 2021.
- [319] X. Wang and L. Tong. Innovations autoencoder and its application in one-class anomalous sequence detection. *Journal of Machine Learning Research*, 23(49):1–27, 2022.
- [320] Y. Wang, J. Hu, X. Chen, and L. Wang. Distributed Bandit Learning: Near-Optimal Regret with Efficient Communication. In *Proceedings of the 7th International Conference on Learning Representations, ICLR*, 2019.
- [321] Y. Wang, Y. Xu, Q. Shi, and T. H. Chang. Quantized Federated Learning under Transmission Delay and Outage Constraints. *IEEE Journal on Selected Areas in Communications*, 40(1):323–341, 2022.
- [322] Z. Wang and N. de Freitas. Theoretical analysis of bayesian optimisation with unknown gaussian process hyper-parameters, 2014.
- [323] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–367, 2016.
- [324] Z. Wang and S. Jegelka. Max-value entropy search for efficient Bayesian optimization. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 7, pages 5530–5543, 2017.
- [325] Z. Wang, B. Kim, and L. P. Kaelbling. Regret bounds for meta bayesian optimization with an unknown gaussian process prior. In *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems*, 2018.
- [326] Z. Wang, B. Shakibi, L. Jin, and N. De Freitas. Bayesian multi-scale optimistic optimization. *Journal of Machine Learning Research*, 33:1005–1014, 2014.
- [327] V. Wild, M. Kanagawa, and D. Sejdinovic. Connections and equivalences between the nystr\” om method and sparse variational gaussian processes, 2021.
- [328] B. Woodworth, K. K. Patel, and N. Srebro. Minibatch vs local SGD for

- heterogeneous distributed learning. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, volume 2020-Decem, 2020.
- [329] B. Woodworth, K. K. Patel, S. U. Stich, Z. Dai, B. Bullins, H. Brendan McMahan, O. Shamir, and N. Srebro. Is local SGD better than minibatch SGD? In *Proceedings of the 37th International Conference on Machine Learning, ICML*, pages 10265–10274, 2020.
  - [330] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.
  - [331] Y. Xu and W. Yin. Block stochastic gradient iteration for convex and non-convex optimization. *SIAM Journal on Optimization*, 25(3):1686–1716, 2015.
  - [332] J. Yang, W. Hu, J. D. Lee, and S. S. Du. Impact of representation learning in linear bandits. In *Proceedings of the 9th International Conference on Learning Representations, ICLR*, 2021.
  - [333] L. Yang. Active learning with a drifting distribution. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, pages 2079–2087, 2011.
  - [334] H. Yu, R. Jin, and S. Yang. On the Linear Speedup Analysis of Communication Efficient Momentum SGD for Distributed Non-Convex Optimization. *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, 2019-June:12431–12467, 2019.
  - [335] Y. Yu, J. Wu, and J. Huang. Exploring fast and communication-efficient algorithms in large-scale distributed networks. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 674–683. PMLR, 2020.
  - [336] A. Zhang and Q. Gu. Accelerated stochastic block coordinate descent with optimal sampling. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-Augu, pages 2035–2044, 2016.
  - [337] C. Zhang. Efficient active learning of sparse halfspaces. In *Proceedings of the 31st Annual Conference on Learning Theory, COLT*, pages 1856–1880. PMLR, 2018.
  - [338] C. Zhang and K. Chaudhuri. Beyond Disagreement-based Agnostic Ac-

- tive Learning. *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*, 1(January):442–450, 2014.
- [339] J. Zhang, S. P. Karimireddy, A. Veit, S. Kim, S. J. Reddi, S. Kumar, and S. Sra. Why ADAM Beats SGD for Attention Models, 2019.
- [340] T. Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17(9):2077–2098, 2005.
- [341] W. Zhang, D. Zhou, L. Li, and Q. Gu. Neural Thompson sampling. In *Proceedings of the 9th International Conference on Learning Representations, ICLR*, 2021.
- [342] H. Zhao, B. Li, Z. Li, P. Richtárik, and Y. Chi. BEER: Fast  $O(1/T)$  Rate for Decentralized Nonconvex Optimization with Communication Compression. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems*, 2022.
- [343] T. Zhao, M. Yu, Y. Wang, R. Arora, and H. Liu. Accelerated mini-batch randomized block coordinate descent method. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*, volume 4, pages 3329–3337, 2014.
- [344] Z. Zhao, Y. Mao, Y. Liu, L. Song, Y. Ouyang, X. Chen, and W. Ding. Towards efficient communications in federated learning: A contemporary survey. *Journal of the Franklin Institute*, 2023.
- [345] S. Zheng, C. Shen, and X. Chen. Design and Analysis of Uplink and Downlink Communications for Federated Learning. *IEEE Journal on Selected Areas in Communications*, 39(7):2150–2167, 2021.
- [346] D. Zhou, L. Li, and Q. Gu. Neural contextual bandits with UCB-based exploration. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, pages 11492–11502. PMLR, 2020.
- [347] Z. Zhu, J. Zhu, J. Liu, and Y. Liu. Federated Bandit: A Gossiping Approach. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(1):1–29, 2021.

APPENDIX A  
CHAPTER 2

### A.1 Proof of Theorem 2.4.1

We first give a proof for Theorem 2.4.1 using the two lemmas. Proofs for the lemmas then follow.

We arrive at Theorem 2.4.1 by bounding separately the two terms  $R_1$  and  $R_2$  in the regret decomposition in (2.12) for an arbitrary objective function  $f \in \mathcal{F}_{\alpha,\beta}$ . Note that the consistency/efficiency level  $p$  of an algorithm as defined in Eqn. (2.9) and Eqn. (2.10) is with respect to the worst-case objective function. This implies that when a  $p$ -consistent low-dimensional algorithm is employed for CM, the convergence rates along different coordinates may vary, depending on the reduction of  $f$  to the specific coordinate. Let  $p_k \geq p$  be the convergence rate in the coordinate  $i_k$  chosen in the  $k$ -th iteration given  $\mathbf{x}_{-i_k}^{(k-1)}$ . More specifically, the error with respect to the local minimum  $\mathbf{x}_{i_k, \mathbf{x}^{(k-1)}}^*$  in the  $i_k$ -th coordinate decays as follows.

$$\left( \mathbb{E}[f(x_{i_k, T}, \mathbf{x}_{-i_k}^{(k-1)})] - f(\mathbf{x}_{i_k, \mathbf{x}^{(k-1)}}^*) \right) \sim \Theta(T^{-p_k}), \quad (\text{A.1})$$

where  $x_{i_k, T}$  denotes the one-dimensional query point at time  $T$ .

We start by bounding  $R_2$ .

$$\begin{aligned}
R_2 &\leq \mathbb{E} \left[ \sum_{k=1}^K \sum_{t=t_{k-1}+1}^{t_k} \left[ F(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*, \xi_t) - F(\mathbf{x}^*, \xi_t) \right] \right] \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K \sum_{s=1}^{\tau(\epsilon_k)} \left[ F(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*, \xi_s) - F(\mathbf{x}^*, \xi_s) \right] \right] \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K \mathbb{E} \left[ \sum_{s=1}^{\tau(\epsilon_k)} \left[ F(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*, \xi_s) - F(\mathbf{x}^*, \xi_s) \right] \middle| \mathbf{x}^{(k-1)} \right] \right] \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K [f(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*) - f(\mathbf{x}^*)] \mathbb{E}[\tau(\epsilon_k)] \right] \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K [f(\mathbf{x}^{(k-1)}) - f(\mathbf{x}^*)] \mathbb{E}[\tau(\epsilon_k)] \right] \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K c_2 \gamma^k \epsilon_k^{-1/p_k} \right] \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K c'_2 \left( \frac{1}{\gamma} \right)^{k \frac{1-p_k}{p_k}} \right]
\end{aligned} \tag{A.2}$$

where the fourth line follows from Wald's Identity and  $c_2, c'_2 > 0$  are constants independent of  $T$ . To upper bound the expression obtained in Eqn. (A.2), we use that the total number of samples taken would be upper bounded by the length of the horizon.

$$\sum_{k=1}^K \mathbb{E}[\tau(\epsilon_k)] \leq T. \tag{A.3}$$

Therefore, for some constant  $c_3 > 0$  and independent of  $T$ , we have,

$$\sum_{k=1}^K \left( \frac{1}{\gamma} \right)^{\frac{k}{p_k}} \leq c_3 T. \tag{A.4}$$

Now using Jensen's inequality, we can write,

$$\begin{aligned}
\frac{1}{K} \sum_{k=1}^K \left( \frac{1}{\gamma} \right)^{\frac{k}{p_k}} &\leq \left( \frac{1}{K} \sum_{k=1}^K \left( \frac{1}{\gamma} \right)^{\frac{k}{p_k}} \right)^{1-p}, \\
\implies \sum_{k=1}^K \left( \frac{1}{\gamma} \right)^{\frac{k}{p_k}} &\leq c'_3 T^{1-p} K^p.
\end{aligned} \tag{A.5}$$

for some constant  $c'_3 > 0$ . Note that the expression here is similar to the one obtained in Eqn. (A.2) and in fact can be used to upper bound  $R_2$ .

$$\begin{aligned} R_2 &\leq c'_2 \mathbb{E} \left[ \sum_{k=1}^K \left( \frac{1}{\gamma} \right)^{k \frac{1-p_k}{p_k}} \right] \\ &\leq c'_2 \mathbb{E} \left[ \sum_{k=1}^K \left( \frac{1}{\gamma} \right)^{k \frac{1-p}{p_k}} \right] \\ &\leq c''_2 \mathbb{E} [T^{1-p} K^p] \\ &\leq c''_2 T^{1-p} \mathbb{E} [K]^p \end{aligned}$$

where  $c''_2 > 0$  is a constant independent of  $T$  and the second step is obtained by noting  $p_k \geq p$ . Using the result from Lemma 2.4.2 and plugging it in the above equation, we can conclude that  $R_2$  is  $O(T^{1-p} \log^p T)$ . Note that for  $p = 1$  this boils down to  $O(\log T)$  as required.

We now consider  $R_1$ .

$$\begin{aligned} R_1 &= \mathbb{E} \left[ \sum_{k=1}^K \sum_{t=t_{k-1}+1}^{t_k} \left[ F(\mathbf{x}_t, \xi_t) - F(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*, \xi_t) \right] \right] \\ &= \mathbb{E} \left[ \sum_{k=1}^K \sum_{s=1}^{\tau(\epsilon_k)} \left[ F(\mathbf{x}_{s+t_{k-1}}, \xi_{s+t_{k-1}}) - F(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*, \xi_{s+t_{k-1}}) \right] \right] \\ &= \mathbb{E} \left[ \sum_{k=1}^K \mathbb{E} \left[ \sum_{s=1}^{\tau(\epsilon_k)} \left[ F(\mathbf{x}_{s+t_{k-1}}, \xi_{s+t_{k-1}}) - F(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*, \xi_{s+t_{k-1}}) \right] \middle| \tau(\epsilon_k) \right] \right] \end{aligned}$$

Next we upper bound the above term separately for  $p$ -consistent and efficient

routines. For  $p$ -consistent ( $p < 1$ ) routines, we have,

$$\begin{aligned} R_1 &\leq \mathbb{E} \left[ \sum_{k=1}^K \mathbb{E} \left[ \sum_{s=1}^{\tau(\epsilon_k)} \frac{c_1}{s^{p_k}} \middle| \tau(\epsilon_k) \right] \right] \\ &\leq \mathbb{E} \left[ \sum_{k=1}^K c_1 \mathbb{E} [(\tau(\epsilon_k))^{1-p_k}] \right] \\ &\leq \mathbb{E} \left[ \sum_{k=1}^K c_1 \mathbb{E} [\tau(\epsilon_k)]^{1-p_k} \right] \end{aligned} \quad (\text{A.6})$$

$$\begin{aligned} &\leq \mathbb{E} \left[ \sum_{k=1}^K c'_1 \epsilon_k^{\frac{p_k-1}{p_k}} \right] \\ &\leq \mathbb{E} \left[ \sum_{k=1}^K c''_1 \left( \frac{1}{\gamma} \right)^{\frac{1-p_k}{p_k}} \right] \end{aligned} \quad (\text{A.7})$$

where  $c_1, c'_1, c''_1 > 0$  are all constants independent of  $T$  and Eqn. (A.6) follows from Jensen's inequality. Note that the term obtained in Eqn. (A.7) is of the same order as the one obtained in Eqn. (A.2). Therefore, using the same analysis as in the case of  $R_2$ , we can conclude that  $R_1$  is also  $O(T^{1-p} \log^p T)$  for  $p$ -consistent ( $p < 1$ ) routines. Now for efficient routines we have  $p_k = 1$  for all  $k$ . Along with the efficiency in leveraging the favorable initial conditions, we have

$$\begin{aligned} R_1 &\leq \mathbb{E} \left[ \sum_{k=1}^K \mathbb{E} \left[ \left( f(\mathbf{x}^{(k-1)}) - f(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*) \right)^\lambda \sum_{s=1}^{\tau(\epsilon_k)} \frac{b_2}{s} \middle| \tau(\epsilon_k) \right] \right] \\ &\leq \mathbb{E} \left[ \sum_{k=1}^K b'_2 \gamma^{(k-1)\lambda} \mathbb{E} [\log(\tau(\epsilon_k))] \right] \\ &\leq \mathbb{E} \left[ \sum_{k=1}^K b''_2 (\epsilon_0 \gamma^k)^\lambda \log(\mathbb{E}[\tau(\epsilon_k)]) \right] \end{aligned} \quad (\text{A.8})$$

$$\begin{aligned} &\leq \mathbb{E} \left[ \sum_{k=1}^K b''_2 \epsilon_k^\lambda \log \left( \frac{b_3}{\epsilon_k} \right) \right] \\ &\leq \mathbb{E} \left[ \sum_{k=1}^K b''_2 \left( \epsilon_k^\lambda \log \left( \frac{1}{\epsilon_k} \right) + \log(b_3) \epsilon_k^\lambda \right) \right] \\ &\leq \mathbb{E} \left[ \sum_{k=1}^K b''_2 \left( \frac{1}{\lambda e} + \log(b_3) \epsilon_0^\lambda \right) \right] \end{aligned} \quad (\text{A.9})$$

$$\leq b_4 \mathbb{E}[K] \quad (\text{A.10})$$

where  $b_2, b'_2, b''_2, b_3, b_4 > 0$  are constants independent of  $T$  and Eqn. (A.8) and Eqn. (A.9) are respectively obtained by Jensen's inequality and the fact that  $-x^\lambda \log(x)$  is uniformly upper bounded by  $(\lambda e)^{-1}$  for all  $x > 0$  and for all  $\lambda > 0$ . Using the upper bound on  $\mathbb{E}[K]$  given from Lemma 2.4.2 leads to the  $O(\log T)$  order of  $R_1$  for efficient algorithms. Combining the above bounds on  $R_1$  and  $R_2$ , we arrive at the theorem.

### A.1.1 Proof of Lemma 2.4.2

Note that the first  $K - 1$  iterations are complete by the end of the horizon of length  $T$ . We thus have, for some constants  $b_1, b'_1 > 0$ ,

$$\begin{aligned} T &\geq \mathbb{E} \left[ \sum_{k=1}^{K-1} \mathbb{E}[\tau(\epsilon_k)] \right] \\ &\geq \mathbb{E} \left[ \sum_{k=1}^{K-1} b_1 \epsilon_k^{-1/p_k} \right] \\ &\geq \mathbb{E} \left[ \sum_{k=1}^{K-1} b_1 \epsilon_k^{-1} \right] \\ &\geq \mathbb{E} \left[ \sum_{k=1}^{K-1} b'_1 \gamma^{-k} \right] \\ &\geq \mathbb{E} \left[ b'_1 \frac{\gamma^{-K} - \gamma^{-1}}{\gamma^{-1} - 1} \right] \end{aligned} \tag{A.11}$$

(A.12)

Therefore, we have that  $\mathbb{E} \left[ \left( \frac{1}{\gamma} \right)^K \right] \leq \frac{T(1 - \gamma^{-1})}{b'_1} + \gamma^{-1}$ . Taking logarithms on both sides and then applying Jensen's inequality, we obtain  $\mathbb{E}[K] \leq \log_{\gamma^{-1}} \left( \frac{T(1 - \gamma^{-1})}{b'_1} + \gamma^{-1} \right)$  as required.

### A.1.2 Proof of Lemma 2.4.3

The main idea of the proof revolves around the use of proximal operators which is similar to the convergence analysis in [159]. Specifically, for  $f = \psi + \phi$ , define

$$\mathcal{D}_\phi(\mathbf{x}, \rho) := -2\rho \min_{\mathbf{y} \in \mathcal{X}} \left[ \langle \nabla \psi(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\rho}{2} \|\mathbf{y} - \mathbf{x}\|^2 + \phi(\mathbf{y}) - \phi(\mathbf{x}) \right]. \quad (\text{A.13})$$

Let us assume that we take a step of length  $z_{i_k}$  along a fixed coordinate  $i_k$ . Therefore, using smoothness of  $\nabla \psi$  and separability of  $\phi$ , we can write,

$$f(\mathbf{x}^{(k-1)} + z_{i_k} \mathbf{e}_{i_k}) \leq f(\mathbf{x}) + z_{i_k} [\nabla \psi(\mathbf{x})]_{i_k} + \frac{\beta}{2} z_{i_k}^2 + \phi_{i_k}(x_{i_k} + z_{i_k}) - \phi_{i_k}(x_{i_k}). \quad (\text{A.14})$$

Let  $z_{i_k}$  be such that

$$z_{i_k} = \arg \min_t \left[ t [\nabla \psi(\mathbf{x})]_{i_k} + \frac{\beta}{2} t^2 + \phi_{i_k}(x_{i_k} + t) - \phi_{i_k}(x_{i_k}) \right] \quad (\text{A.15})$$

Using the precision guaranteed by the termination rule and conditioning on  $\mathbf{x}^{(k-1)}$ , we have

$$\begin{aligned} \mathbb{E}[F(\mathbf{x}^k) | \mathbf{x}^{(k-1)}] &\leq f(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*) + \epsilon_k \\ &\leq f(\mathbf{x}^{(k-1)} + z_{i_k} \mathbf{e}_{i_k}) + \epsilon_k \end{aligned} \quad (\text{A.16})$$

Taking expectation over the coordinate index  $i_k$ , which is uniformly distributed over the set  $\{1, 2, \dots, d\}$ , we can write,

$$\begin{aligned}
\mathbb{E}[F(\mathbf{x}^k) | \mathbf{x}^{(k-1)}] &\leq \mathbb{E}_{i_k}[f(\mathbf{x}^{(k-1)} + z_{i_k} \mathbf{e}_{i_k})] + \epsilon_k, \\
&\leq \mathbb{E}_{i_k}\left[f(\mathbf{x}^{(k-1)}) + z_{i_k}[\nabla \psi(\mathbf{x}^{(k-1)})]_{i_k} + \frac{\beta}{2}z_{i_k}^2 2 + \phi_{i_k}(x_{i_k}^{(k-1)} + z_{i_k}) - \phi_{i_k}(x_{i_k}^{(k-1)})\right] + \epsilon_k, \\
&\leq f(\mathbf{x}^{(k-1)}) + \frac{1}{d} \sum_{i=1}^d \min_{t_i} \left[ t_i [\nabla \psi(\mathbf{x}^{(k-1)})]_i + \frac{\beta}{2} t_i^2 2 + \phi_i(x_i^{(k-1)} + t_i) - \phi_i(x_i^{(k-1)}) \right] + \epsilon_k, \\
&\leq f(\mathbf{x}^{(k-1)}) + \frac{1}{d} \min_{\mathbf{t}} \left[ \sum_{i=1}^d t_i [\nabla \psi(\mathbf{x}^{(k-1)})]_i + \frac{\beta}{2} t_i^2 2 + \phi_i(x_i^{(k-1)} + t_i) - \phi_i(x_i^{(k-1)}) \right] + \epsilon_k, \\
&\leq f(\mathbf{x}^{(k-1)}) + \frac{1}{d} \min_{\mathbf{y}} \left[ \langle F_1(\mathbf{x}^{(k-1)}), \mathbf{y} - \mathbf{x}^{(k-1)} \rangle + \frac{\beta}{2} \|\mathbf{y} - \mathbf{x}^{(k-1)}\|^2 + \phi(\mathbf{y}) - \phi(\mathbf{x}^{(k-1)}) \right] + \epsilon_k, \\
&\leq f(\mathbf{x}^{(k-1)}) - \frac{1}{2d\beta} \mathcal{D}_g(\mathbf{x}^{(k-1)}, \beta) + \epsilon_k, \\
&\leq f(\mathbf{x}^{(k-1)}) - \frac{\alpha}{d\beta} (f(\mathbf{x}^{(k-1)}) - f(\mathbf{x}^*)) + \epsilon_k
\end{aligned} \tag{A.17}$$

where the step uses the proximal PL inequality for strongly convex functions described in [159]. Taking expectation over  $\mathbf{x}^{(k-1)}$ , we obtain,

$$\mathbb{E}[F(\mathbf{x}^k)] - f(\mathbf{x}^*) \leq (\mathbb{E}[F(\mathbf{x}^{(k-1)})] - f(\mathbf{x}^*)) \left(1 - \frac{\alpha}{d\beta}\right) + \epsilon_0 \gamma^k \tag{A.18}$$

Let  $\phi_k = \mathbb{E}[F(\mathbf{x}^k)] - f(\mathbf{x}^*)$ . We claim that  $\phi_k \leq F_0 \gamma^k$  where  $F_0 = \max \left\{ f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*), \frac{\epsilon_0}{(1-\gamma)} \right\}$ . This can be proved using induction. For the base case, we have  $\phi_0 = f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*) \leq F_0$  by definition. Assume it is true for  $k-1$ , then we have,

$$\begin{aligned}
\phi_k &\leq \left(1 - \frac{\alpha}{d\beta}\right) \phi_{k-1} + \epsilon_0 \gamma^k \\
&\leq \gamma^2 (F_0 \gamma^{k-1}) + \epsilon_0 \gamma^k \\
&\leq F_0 \gamma^{k+1} + \epsilon_0 \gamma^k \\
&\leq \gamma^k (F_0 \gamma + \epsilon_0) \\
&\leq F_0 \gamma^k.
\end{aligned} \tag{A.19}$$

The last step follows from the choice of  $F_0$ . This completes the proof.

## A.2 Proof of Lemma 2.5.1

We first prove the efficiency of SGD followed by the order optimality of the termination rule.

### A.2.1 Efficiency of the SGD Routine

Consider a one-dimensional stochastic function  $F(x)$  with stochastic gradient given by  $G(x)$ . Let  $x^*$  be the minimizer of the function, i.e.,  $x^* = \arg \min_{x \in \mathcal{X}} f(x)$  where  $f(x) = \mathbb{E}[F(x)]$  and  $\mathcal{X}$  is the domain of the function. The iterates generated by SGD with initial point  $x_0$  satisfy the following relation,

$$\begin{aligned}
\mathbb{E} [\|x_{t+1} - x^*\|^2] &= \mathbb{E} [\|\text{proj}_{\mathcal{X}}(x_t - \eta_t G(x_t) - x^*)\|^2] \\
&\leq \mathbb{E} [\|x_t - \eta_t G(x_t) - x^*\|^2] \\
&\leq \mathbb{E} [\|x_t - x^*\|^2 - 2\eta_t \langle G(x_t), x_t - x^* \rangle + \eta_t^2 \|G(x_t)\|^2] \\
&\leq \mathbb{E} [\|x_t - x^*\|^2] - 2\eta_t \mathbb{E} [\alpha \|x_t - x^*\|^2] + \eta_t^2 \mathbb{E} [\|G(x_t)\|^2] \\
&\leq (1 - 2\eta_t \alpha) \mathbb{E} [\|x_t - x^*\|^2] + \eta_t^2 g_{\max}^2
\end{aligned} \tag{A.20}$$

Next we show that the iterates satisfy  $\mathbb{E} [\|x_t - x^*\|^2] \leq \frac{\mu_0}{1 + vt}$  for all  $t \geq 0$  based on an inductive argument. The base case is ensured by choosing  $\mu_0$  satisfying  $\mu_0 \geq \mathbb{E}[\|x_0 - x^*\|^2]$ . For the induction step, note that the stepsizes are chosen as  $\eta_t = \frac{\mu}{1 + vt}$  with  $\mu = \frac{\mu_0 \alpha}{2g_{\max}^2}$  and  $v = \frac{\mu_0 \alpha^2}{4g_{\max}^2}$ . We continue with Eqn. (A.20) as

follows.

$$\begin{aligned} \mathbb{E} [\|x_{t+1} - x^*\|^2] &\leq (1 - 2\eta_t \alpha) \mathbb{E} [\|x_t - x^*\|^2] + \eta_t^2 g_{\max}^2 \\ &\leq \left(1 - 2\frac{\mu\alpha}{1+\nu t}\right) \frac{\mu_0}{1+\nu t} + \frac{\mu^2}{(1+\nu t)^2} g_{\max}^2 \\ &\leq \frac{\mu_0}{1+\nu(t+1)} + \left(\frac{\mu_0}{1+\nu t} - \frac{\mu_0}{1+\nu(t+1)}\right) + \frac{\mu}{(1+\nu t)^2} (\mu g_{\max}^2 - 2\mu_0 \alpha) \\ &\leq \frac{\mu_0}{1+\nu(t+1)} + \frac{\mu_0^2}{(1+\nu t)^2} (\mu^2 g_{\max}^2 - 2\mu\mu_0\alpha + \mu_0\nu) \end{aligned} \quad (\text{A.21})$$

$$\leq \frac{\mu_0}{1+\nu(t+1)} + \frac{\mu_0^2}{(1+\nu t)^2} \left( \frac{\mu_0^2 \alpha^2}{4g_{\max}^4} g_{\max}^2 - \frac{\mu_0^2 \alpha^2}{g_{\max}^2} + \frac{\mu_0^2 \alpha^2}{4g_{\max}^2} \right) \quad (\text{A.22})$$

$$\leq \frac{\mu_0}{1+\nu(t+1)} \quad (\text{A.23})$$

Therefore, the iterates generated by SGD satisfy  $\mathbb{E} [\|x_t - x^*\|^2] \leq \frac{\mu_0}{1+\nu t}$  for all  $t \geq 0$ .

To ensure efficiency with respect to the initial point  $x_0$ ,  $\mu_0$  should be of the order  $\mu_0 \leq C\mathbb{E}[\|x_0 - x^*\|^2]$  as  $x_0$  goes to  $x^*$  for some  $C > 0$  (see below how this can be ensured within the PCM framework). Based on the strong convexity and smoothness of the function, the condition on the iterates can be translated to a condition on the function values as given below

$$\mathbb{E}[F(x_t) - f(x^*)] \leq \frac{\beta C}{\alpha} \frac{\mathbb{E}[f(x_0) - f(x^*)]}{1+\nu t}, \quad (\text{A.24})$$

which implies that SGD is an efficient policy with  $\lambda = 1$ .

For implementation in PCM, the choice of  $\mu_0$  can be simplified using the relation on the CM iterates outlined in Lemma 2.4.3. In iteration  $k$ ,  $\mathbf{x}^{(k-1)}$  is the initial point, therefore, we can write,  $\mathbb{E} [\|\mathbf{x}^{(k-1)} - \mathbf{x}_{(i_k, x^{(k-1)})}^*\|^2] \leq \frac{2}{\alpha} \mathbb{E} [f(\mathbf{x}^{(k-1)}) - f(\mathbf{x}_{(i_k, x^{(k-1)})}^*)] \leq \mathbb{E} [f(\mathbf{x}^{(k-1)}) - f(\mathbf{x}^*)] \leq F_0 \gamma^{k-1}$ . Thus, for an appropriate choice of  $\mu_0$  for the first iteration, its value for consequent iterations can be obtained by the relation  $\mu_0(k) = \gamma \mu_0(k-1)$ , where  $\mu_0(k)$  is the value of  $\mu_0$  used in iteration  $k$ .

## A.2.2 Order Optimality of the Termination Rule

The correctness of the termination rule follows in a straightforward manner from the relation obtained on the iterates in the previous part. Using smoothness of the function and the relation obtained in Eqn. (A.23), we have,  $\mathbb{E}[F(x_t) - f(x^*)] \leq \frac{\mu_0\beta}{2(1 + vt)}$ . Let  $t_0$  be such that,  $\frac{\mu_0\beta}{2(1 + vt_0)} \leq \epsilon$ . On rearranging this equation, we obtain  $t_0 \geq \frac{\mu_0\beta}{2\epsilon\nu} - \frac{1}{\nu}$ . Therefore, for all  $t \geq t_0$ , we have  $\mathbb{E}[F(x_t) - f(x^*)] \leq \epsilon$ . Since our choice of termination rule satisfies the above condition, we can conclude that our termination rule ensures the required precision. The order optimality of the termination also follows directly from the expression.

## A.3 Random Walk on a Tree for PCM

In this section, we analyze the performance of the Random Walk on a Tree (RWT) under the PCM setup. The analysis is split into four subsections. The first two subsections analyse of the termination rule and the expected number of steps taken in one run of RWT. In the third subsection, we bound the regret incurred in one iteration of RWT and then use it obtain the final regret bound on RWT in the fourth subsection.

### A.3.1 Termination Rule

We begin with a lemma that states the correctness of the termination rule and also relate the expected second moment of the gradient of the final point

to the required precision  $\epsilon$ . Recall that the termination rule specified that if at a certain point the number of samples taken in a sequential test exceeds  $N_0(\epsilon) = \frac{40\sigma_0^2}{\alpha\epsilon} \log\left(\frac{2}{\check{p}} \log\left(\frac{80\sigma_0^2}{\alpha\check{p}\epsilon}\right)\right)$  the algorithm must terminate, returning the current point being probed.

**Lemma A.3.1.** *Let  $x_{\tau(\epsilon)}$  denote the final point obtained under the termination rule. Then we have the following relations  $\mathbb{E}[f(x_{\tau(\epsilon)}) - f(x^*)] \leq \epsilon$  and  $\mathbb{E}[g^2(x_{\tau(\epsilon)})] \leq 2\alpha\epsilon$ .*

*Proof.* The first part of the lemma directly follows from the second part using the strong convexity of the function. Since  $f$  is strongly convex, we have  $\mathbb{E}[f(x_{\tau(\epsilon)}) - f(x^*)] \leq \frac{1}{2\alpha}\mathbb{E}[g^2(x_{\tau(\epsilon)})] \leq \epsilon$  as required. Hence, we just focus on the proving the bound on the gradient.

To obtain the bound on the gradient, we leverage the primary idea underlying the design of the threshold in the termination rule. The threshold is designed to ensure that the gradient at the point at which the algorithm terminates is sufficiently small with high probability. We use this high probability bound to obtain the required bound on the second moment of the gradient.

Define  $\rho := \frac{\alpha\epsilon}{2}$ . We claim that under the given termination rule,  $|g(x_{\tau(\epsilon)})| \leq \rho$  holds with high probability. To prove the claim, we consider the probability that the random number of samples taken in a sequential test, denoted by  $\hat{T}$ , exceed

any number  $n$ . For any point with  $g(x) > 0$ , we have,

$$\begin{aligned}
\mathbb{P}[\hat{T} > n] &\leq \mathbb{P}\left[\forall s \leq n : \bar{G}_s(x) + \sqrt{\frac{5\sigma_0^2}{s} \log\left(\frac{6 \log s}{\sqrt{\check{p}}}\right)} > 0, \text{ and } \bar{G}_s - \sqrt{\frac{5\sigma_0^2}{s} \log\left(\frac{6 \log s}{\sqrt{\check{p}}}\right)} < 0\right], \\
&\leq \mathbb{P}\left[\forall s \leq n : \bar{G}_s - \sqrt{\frac{5\sigma_0^2}{s} \log\left(\frac{6 \log s}{\sqrt{\check{p}}}\right)} < 0\right], \\
&\leq \mathbb{P}\left[\bar{G}_n - \sqrt{\frac{5\sigma_0^2}{n} \log\left(\frac{6 \log n}{\sqrt{\check{p}}}\right)} < 0\right], \\
&\leq \mathbb{P}\left[\bar{G}_n - \mathbb{E}_\xi[G(x, \xi)] < \sqrt{\frac{5\sigma_0^2}{n} \log\left(\frac{6 \log n}{\sqrt{\check{p}}}\right)} - g(x)\right], \\
&\leq \exp\left(-\frac{n}{2\sigma_0^2} \left(\sqrt{\frac{5\sigma_0^2}{n} \log\left(\frac{6 \log n}{\sqrt{\check{p}}}\right)} - g(x)\right)^2\right).
\end{aligned} \tag{A.25}$$

The threshold  $N_0(\epsilon)$  can be equivalently written in terms of  $\rho$  as  $s_0(\rho) = \frac{20\sigma_0^2}{\rho^2} \log\left(\frac{2}{\check{p}} \log\left(\frac{40\sigma_0^2}{\check{p}\rho^2}\right)\right)$ . For all  $n > s_0(\rho)$ , we have,

$$\mathbb{P}[\hat{T} > n] \leq \exp\left(-\frac{n}{2\sigma_0^2} \left(\frac{\rho}{2} - g(x)\right)^2\right) \tag{A.26}$$

This can be obtained by plugging  $n = s_0(\rho)$  in the upper bound in Eqn. (A.25). A more detailed analysis of this step can be found in Appendix B in [304]. A similar analysis can be carried out for any point with  $g(x) < 0$ . Using Eqn. (A.26), we can conclude that if the number of samples in a local test at a point  $x$  exceed  $s_0(\rho)$ , then  $|g(x)| \leq \rho$  with probability at least  $1 - \delta_0$  where  $\delta_0 = \exp\left(-\frac{s_0(\rho)\rho^2}{8\sigma_0^2}\right)$ .

We can now bound the second moment of the gradient as follows by noting

that  $\delta_0 \leq 1/2$

$$\begin{aligned}
\mathbb{E}[g^2(x_{\tau(\epsilon)})] &\leq \rho^2 \mathbb{P}[g(x_{\tau(\epsilon)}) \leq \rho] + \mathbb{E}[g^2(x_{\tau(\epsilon)}) \mathbb{I}_{\{g(x_{\tau(\epsilon)}) > \rho\}}], \\
&\leq \rho^2 + \sum_{r=1}^{\infty} (r+1)^2 \rho^2 \mathbb{P}(r\rho < g(x_{\tau(\epsilon)}) \leq (r+1)\rho), \\
&\leq \rho^2 + \sum_{r=1}^{\infty} (r+1)^2 \rho^2 \mathbb{P}(g(x_{\tau(\epsilon)}) > r\rho), \\
&\leq \rho^2 + \sum_{r=1}^{\infty} (r+1)^2 \rho^2 \exp\left(-\frac{s_0(\rho)}{2\sigma_0^2} \left(\frac{\rho}{2} - r\rho\right)^2\right), \\
&\leq \rho^2 + \sum_{r=1}^{\infty} (r+1)^2 \rho^2 \exp\left(-\frac{s_0(\rho)\rho^2}{8\sigma_0^2} (2r-1)^2\right), \\
&\leq \rho^2 + \sum_{r=1}^{\infty} (r+1)^2 \rho^2 \delta_0^{(2r-1)^2}, \\
&\leq \rho^2 + 2.01\rho^2, \\
&\leq 4\rho^2.
\end{aligned} \tag{A.27}$$

By plugging in  $\rho = \sqrt{\frac{\alpha\epsilon}{2}}$ , we arrive at the required result.  $\square$

As it is easier to analyze expressions in terms of the gradient and not the function values, we will use the expressions in terms of  $\rho$  for the rest of the section keeping in mind its relation with the required precision of  $\epsilon$ .

### A.3.2 Upper Bound on $\mathbb{E}[\tau(\epsilon)]$

To bound the expected number of samples taken in one iteration of the PCM-RWT algorithm with precision  $\epsilon$ ,  $\mathbb{E}[(\tau(\epsilon))]$ , we need to obtain a bound on the number of steps taken by the random walk before termination. The bound on  $\mathbb{E}[\tau(\epsilon)]$  follows by noting that the number of samples taken in each sequential test before termination is bounded by the threshold specified in the termination

rule. For the bound on the number of steps in the random walk, we note that as the random walk gets to a deeper level in the tree, the magnitude of the gradient reduces. Consequently, the probability that the number of samples taken in the sequential test will cross the threshold increases as the walk goes to a deeper level in the tree. These decreasing tail probabilities can then be used to obtain a bound on the expected number of steps in the random walk.

Assume the minima to be  $x^* = 0$ . Such an assumption leads to no loss of generality as the analysis can easily be modified for any point in the interval and for any interval of any length. We begin the analysis for the case when the random walk is initialized at the root node. This analysis can be easily modified to accommodate the initialization at a deeper level.

We divide the tree into a sequence of subtrees given by  $\mathcal{T}_1, \mathcal{T}_2, \dots$  where for all  $i = 1, 2, \dots$ , the subtree  $\mathcal{T}_i$  contains the node corresponding to the interval  $[0, 2^{-(i-1)}]$  and its right child along with all its children. Thus,  $\mathcal{T}_i$ 's are half trees rooted at level  $i - 1$ , along with their root. This construction is similar to the one outlined in [313]. Since the random walk is biased towards the minimizer, therefore given the construction of  $\mathcal{T}_i$ , the probability that random walk is still in one of such subtrees would decrease with time. To formalize this idea, we consider the last passage times of any subtree  $\mathcal{T}_i$ . Let  $\tau_1$  denote the last passage time to  $\mathcal{T}_1$ .

The analysis of the last passage time of  $\mathcal{T}_1$  can be mapped to the problem of a random walk on the set  $S = \{-1, 0, 1, 2, \dots\}$ . The underlying idea is that each non-negative integer can be mapped to the corresponding level in subtree. Our random walk on the tree can between different levels is then equivalent to a random walk on these integers. The equivalence follows by noting that

the specific intervals on any level are all identical as they do not contain the minimizer and thus can be abstracted into a single entity. Hence, we map the root node to 0, and set of nodes at level  $j$  in subtree  $\mathcal{T}_1$  to integer  $j$  for  $j > 0$ . Lastly, we map the left subtree containing all nodes in the interval  $[0, 0.5]$  to  $-1$ , which corresponds to an exit from the subtree  $\mathcal{T}_1$ .

The random walk can be modelled as a Markov chain on the set  $S$ , where  $\mathbb{P}(j \rightarrow j+1) = 1-p$  for all  $j \in S$ ,  $\mathbb{P}(j \rightarrow j-1) = p$  for all  $j \geq 0$  and  $\mathbb{P}(-1 \rightarrow -1) = p$ . The probability  $p = \check{p}^3 > 0.5$  is the probability of moving in correct direction where  $\check{p}$  is the confidence level in the sequential test. The initial state is 0.

Since  $-1$  denotes the state corresponding to exiting the subtree  $\mathcal{T}_1$ , therefore our random walk still being in  $\mathcal{T}_1$  after  $n$  steps is the same as the Markov Chain being in a state  $j$  for  $j \geq 0$  after  $n$  steps. Furthermore, since the Markov Chain was initialized at 0, therefore being in state  $j \geq 0$  implies that the number of steps taken in the positive direction are at least as many as those taken in the negative direction. Combining all these ideas along with noting the specific structure of the transition matrix, we can conclude that

$$\mathbb{P}(\tau_1 > n) = \mathbb{P}(Z \leq n/2), \quad (\text{A.28})$$

where  $Z \sim \text{Bin}(n, p)$ . Writing expectation as the sum of tail probabilities,

$$\begin{aligned} \mathbb{E}[\tau_1] &= \sum_{n=0}^{\infty} \mathbb{P}(\tau_1 > n), \\ &= \sum_{n=0}^{\infty} \mathbb{P}(Z \leq n/2), \\ &= \sum_{n=0}^{\infty} \exp(-2(p - 1/2)^2 n), \\ &= \frac{1}{1 - \exp(-2(p - 1/2)^2)}. \end{aligned} \quad (\text{A.29})$$

The third step is obtained using Hoeffding's inequality. We can leverage the symmetry of the random walk and the binary tree to obtain the expected last passage time for any other subtree  $\mathcal{T}_i$ .

Let for all  $i \geq 1$ ,  $N_{\mathcal{T}_i}$  denote the random number of steps taken in subtree  $\mathcal{T}_i$  before exiting that subtree and  $E_i$  denote the event that the random walk does not terminate in tree  $\mathcal{T}_i$ . If  $N_{RW}$  denotes the random number of steps taken by the random walk before termination then

$$\mathbb{E}[N_{RW}] = \mathbb{E}[N_{\mathcal{T}_1}] + \sum_{i=2}^{\infty} \mathbb{P}\left(\bigcap_{j=1}^{i-1} E_j\right) \mathbb{E}\left[N_{\mathcal{T}_i} \middle| \bigcap_{j=1}^{i-1} E_j\right]. \quad (\text{A.30})$$

By definition we have  $\mathbb{E}[N_{\mathcal{T}_1}] = \mathbb{E}[\tau_1]$ . Furthermore, one can note that due to symmetry in the structure of the binary tree,  $\mathbb{E}\left[N_{\mathcal{T}_i} \middle| \bigcap_{j=1}^{i-1} E_j\right] = \mathbb{E}[\tau_1]$ . Hence, to evaluate Eqn. (A.30) we need to find a bound on  $\mathbb{P}\left(\bigcap_{j=1}^{i-1} E_j\right)$ , the probability that the random walk does not terminate in  $\mathcal{T}_j$  for  $j = 1, 2, \dots, i-1$  and  $i \geq 2$ . To bound this probability, consider the event that local sequential test takes less than  $s_0(\rho)$  samples before termination when the magnitude of the gradient of the point being sampled is less than  $\rho$ . Let the event be denoted by  $E_f(\rho)$  and let  $\mathbb{P}(E_f(\rho)) \leq \eta_\rho$  for some  $\eta_\rho < 1$ .

Note that for any level  $i > \log_2(\beta/\rho)$ , the length of the interval at this level would be lesser than  $\rho/\beta$ . Using the smoothness of the function, it follows that the magnitude of gradient of any point probed in  $\mathcal{T}_i$  for  $i > \log_2(\beta/\rho)$  would be lesser than  $\rho$ . Therefore, for every  $i > \log_2(\beta/\rho)$ , if  $E_i$  occurs then  $E_f(\rho)$  would definitely have occurred. Consequently, for all  $i > i_0$ ,

$$\mathbb{P}\left(\bigcap_{j=1}^{i-1} E_j\right) \leq \eta_\rho^{i-i_0}, \quad (\text{A.31})$$

where  $i_0 = \lceil \log_2(\beta/\rho) \rceil$ . For  $i \leq i_0$ , we can crudely upper bound this probability

with 1. Plugging these relations into Eqn. (A.30), we obtain,

$$\begin{aligned}
\mathbb{E}[N_{RW}] &= \mathbb{E}[N_{T_1}] + \sum_{i=2}^{\infty} \mathbb{P}\left(\bigcap_{j=1}^{i-1} E_j\right) \mathbb{E}\left[N_{T_i} \middle| \bigcap_{j=1}^{i-1} E_j\right], \\
&\leq \mathbb{E}[\tau_1] \left( i_0 + 1 + \sum_{i=i_0+1}^{\infty} \eta_{\rho}^{i-i_0} \right), \\
&\leq \frac{1}{1 - \exp(-2(p - 1/2)^2)} \left( \lceil \log_2(\beta/\rho) \rceil + 1 + \sum_{i=1}^{\infty} \eta_{\rho}^i \right), \\
&\leq \frac{1}{1 - \exp(-2(p - 1/2)^2)} \left( \log_2(\beta/\rho) + 2 + \frac{\eta_{\rho}}{1 - \eta_{\rho}} \right). \tag{A.32}
\end{aligned}$$

The above analysis provides an upper bound for the number of steps taken by the random walk when it is initialized at the root node. However, as mentioned previously, we would want the RWT to be initialized at a deeper level in the tree to leverage the favorable initial conditions. We can perform a similar analysis for number of steps taken by random walk in the case when RWT is initialized at a deeper level to leverage the favorable initial conditions.

As given in the description of PCM-RWT, we initialize the algorithm the node which contains the initial point and at a level where the interval length is lesser than  $\sqrt{\log_2\left(\frac{\beta}{\rho}\right) \frac{2\mu_0}{\alpha}}$ , where  $\mu_0$  is a carefully chosen hyperparameter. If the threshold exceeds 1, then we begin at the root. To analyze the number of steps taken by the random walk, we consider the event that  $|x_0 - x^*|^2 \leq \log_2\left(\frac{\beta}{\rho}\right) \frac{2\mu_0}{\alpha}$ , where  $x_0$  is randomly chosen. We denote the event by  $E_{x_0}$ . Under this event, we can carry out a similar analysis as before, with a minor change that instead of  $i_0$ , the maximum depth would be  $i_1 \leq \log_2\left(\sqrt{\log_2\left(\frac{\beta}{\rho}\right) \frac{2\mu_0}{\alpha}} \frac{\beta}{\rho}\right) + 1$ . Under the case the above event does not occur, the random walk would have to take no more than an additional  $i_2 \leq \log_2\left(\sqrt{\log_2\left(\frac{\beta}{\rho}\right) \frac{2\mu_0}{\alpha}}\right) + 1$  steps before the previous analysis is again applicable. If  $N_{RW-new}$  denotes the random number of steps taken by the random walk under this initialization scheme, then on combining

the above results, we can write,

$$\begin{aligned}\mathbb{E}[N_{RW\text{-new}}] &\leq \mathbb{P}(E_{x_0}) \left( \zeta_p \left( \log_2 \left( \sqrt{\log_2 \left( \frac{\beta}{\rho} \right) \frac{2\mu_0}{\alpha}} \frac{\beta}{\rho} \right) + \hat{\eta}_\rho \right) \right), \\ &\quad + \mathbb{P}(E_{x_0}^c) \left( \zeta_p \left( \log_2 \left( \sqrt{\log_2 \left( \frac{\beta}{\rho} \right) \frac{2\mu_0}{\alpha}} \right) + \log_2 \left( \frac{\beta}{\rho} \right) + 1 + \hat{\eta}_\rho \right) \right), \quad (\text{A.33})\end{aligned}$$

where  $E^c$  denotes the complement of an event  $E$ ,  $\zeta_p = \frac{1}{1 - \exp(-2(p - 1/2)^2)}$  and  $\hat{\eta}_\rho = 2 + \frac{\eta_\rho}{1 - \eta_\rho}$ . We can bound  $\mathbb{P}(E_{x_0}^c)$  using Markov's inequality as follows,

$$\begin{aligned}\Pr \left( |x_0 - x^*|^2 > \log_2 \left( \frac{\beta}{\rho} \right) \frac{2\mu_0}{\alpha} \right) &\leq \Pr \left( f(x_0) - f(x^*) > \log_2 \left( \frac{\beta}{\rho} \right) \mu_0 \right), \\ &\leq \mathbb{E} [f(x_0) - f(x^*)] \left( \log_2 \left( \frac{\beta}{\rho} \right) \mu_0 \right)^{-1}. \quad (\text{A.34})\end{aligned}$$

Setting  $\mu_0 = \mathbb{E} [f(x_0) - f(x^*)]$  and plugging Eqn. (A.34) in Eqn. (A.33), we obtain,

$$\begin{aligned}\mathbb{E}[N_{RW\text{-new}}] &\leq \left( \zeta_p \left( \frac{1}{2} \log_2 \left( \log_2 \left( \frac{\beta}{\rho} \right) \frac{2\mathbb{E} [f(x_0) - f(x^*)] \beta^2}{\alpha} \right) \frac{\beta^2}{\rho^2} \right) + \hat{\eta}_\rho \right) \\ &\quad + \left( \log_2 \left( \frac{\beta}{\rho} \right) \right)^{-1} \left( \zeta_p \left( \frac{1}{2} \log_2 \left( \log_2 \left( \frac{\beta}{\rho} \right) \frac{2\mathbb{E} [f(x_0) - f(x^*)]}{\alpha} \right) + \log_2 \left( \frac{\beta}{\rho} \right) + 1 + \hat{\eta}_\rho \right) \right), \\ &\leq \left( \zeta_p \left( \frac{1}{2} \log_2 \left( \log_2 \left( \frac{\beta}{\rho} \right) \frac{2\mathbb{E} [f(x_0) - f(x^*)] \beta^2}{\alpha} \right) \frac{\beta^2}{\rho^2} \right) + \hat{\eta}_\rho \right) \\ &\quad + \zeta_p \left( \frac{\log_2(\beta/g_{\max}) + 0.5 \log_2(2g_{\max}/\alpha) + \hat{\eta}_\rho + 1.5}{\log_2(\beta/g_{\max})} \right). \quad (\text{A.35})\end{aligned}$$

As in the case of SGD, a similar analysis can be carried out for any  $\mu_0 \sim \Theta(E [f(x_0) - f(x^*)])$ . Furthermore, for PCM-RWT,  $\mu_0$  can be tuned for each iteration in the same manner as described for PCM-SGD, that is, by decreasing it by a factor of  $\gamma$  after every iteration. This proof can be readily extended to interval of any length  $l$ , by changing the value of  $i_0$  to  $\log_2(\beta l/\rho)$  and also appropriately changing the bound on  $\mathbb{E} [f(x_0) - f(x^*)]$  in Eqn. (A.35). For a different minimizer, the sequence of subtrees  $\mathcal{T}_i$ 's can be appropriately modified as described in [313] to obtain the same result.

Finally, using Eqn. (A.35), we can obtain the bound on  $\mathbb{E}[\tau(\epsilon)]$ . If  $M_{RW}$  denotes the random number of local tests carried out before termination then  $\mathbb{E}[M_{RW}] \leq \mathbb{E}[3N_{RW-new} + 3]$ . Moreover, since the number of samples in each test can be at most  $s_0(\rho)$ , therefore, the expected number of samples can be no more than  $\mathbb{E}[M_{RW}]s_0(\rho)$ . Substituting the different bounds and the relation between  $\rho$  and  $\epsilon$ , we obtain that for some constant  $\tau_0 > 0$ , independent of  $\epsilon$

$$\mathbb{E}[\tau(\epsilon)] \leq \frac{\tau_0}{\epsilon} \log\left(\frac{\mathbb{E}[f(x_0) - f(x^*)]}{\epsilon}\right) \log^2\left(\log\left(\frac{1}{\epsilon}\right)\right). \quad (\text{A.36})$$

### A.3.3 Regret in One CM Iteration

In this section, we perform a brief analysis of the regret incurred in one CM iteration. This corresponds to the inner sum in the term  $R_1$  in the regret decomposition of PCM, capturing the regret incurred by the routine  $v$  in the local one dimensional minimization. Let  $x_{(m)}$  denote the sampling point at the  $m^{\text{th}}$  time the local test is called by the random walk module and  $\hat{T}_m$  denote the random number of samples taken at this point. Therefore, if  $R_{RWT}(\epsilon)$  denotes the regret incurred by RWT in one CM iteration to get to precision of  $\epsilon$ , then we have,

$$\begin{aligned} R_{RWT}(\epsilon) &= \mathbb{E}\left[\sum_{m=1}^{M_{RW}} \sum_{t=1}^{\hat{T}_m} F(x_{(m)}; \xi_t) - F(x^*, \xi_t)\right], \\ &\leq \mathbb{E}\left[\sum_{m=1}^{M_{RW}} \sum_{t=1}^{\hat{T}_m} \frac{1}{2\alpha} [g(x_{(m)})]^2\right], \\ &\leq \mathbb{E}\left[\sum_{m=1}^{M_{RW}} \mathbb{E}[\hat{T}_m] \frac{1}{2\alpha} [g(x_{(m)})]^2\right]. \end{aligned} \quad (\text{A.37})$$

Note that  $\hat{T}_m$  is the random number of samples taken at sampling point  $x_{(m)}$  with the termination rule. If  $\tilde{T}$  denotes the random number of samples taken without the termination rule then,  $\hat{T}_m = \tilde{T} \mathbb{1}\{\tilde{T} \leq s_0(\rho)\}$  where  $\rho = \sqrt{\alpha\epsilon/2}$ . To

bound  $\mathbb{E}[\hat{T}_m]$ , we use different methods depending on the gradient of the sampling point. If  $|g(x_{(m)})| \leq \rho$ , then we use the trivial bound  $\mathbb{E}[\hat{T}_m] = \mathbb{E}[\tilde{T}] \mathbb{1}\{\tilde{T} \leq s_0(\rho)\} \leq s_0(\rho)$ . For the other case of  $|g(x_{(m)})| > \rho$ , we note that  $\mathbb{E}[\hat{T}_m] \leq \mathbb{E}[\tilde{T}] \leq \frac{40\sigma_0^2}{g(x_{(m)})^2} \log\left(\frac{2}{\check{p}} \log\left(\frac{40\sigma_0^2}{\check{p}g(x_{(m)})^2}\right)\right) + 2 \leq \frac{40\sigma_0^2}{g(x_{(m)})^2} \log\left(\frac{2}{\check{p}} \log\left(\frac{40\sigma_0^2}{\check{p}\rho^2}\right)\right) + 2$ . Plugging these bounds in Eqn. (A.37), we obtain,

$$\begin{aligned}
R_{RWT}(\epsilon) &\leq \mathbb{E} \left[ \sum_{m=1}^{M_{RW}} \frac{g(x_{(m)})^2}{2\alpha} \left( \mathbb{E}[\hat{T}_m] \mathbb{1}\{|g(x_{(m)})| > \rho\} + \mathbb{E}[\hat{T}_m] \mathbb{1}\{|g(x_{(m)})| \leq \rho\} \right) \right], \\
&\leq \mathbb{E} \left[ \sum_{m=1}^{M_{RW}} \left\{ \frac{40\sigma_0^2}{[g(x_{(m)})]^2} \log\left(\frac{2}{\check{p}} \log\left(\frac{40\sigma_0^2}{\check{p}\rho^2}\right)\right) + 2 \right\} \frac{1}{2\alpha} [g(x_{(m)})]^2 \mathbb{1}\{|\mathbb{E}_\xi[G(x_{(m)}; \xi)]| > \rho\} \right. \\
&\quad \left. + \frac{20\sigma_0^2}{\rho^2} \log\left(\frac{2}{\check{p}} \log\left(\frac{40\sigma_0^2}{\check{p}\rho^2}\right)\right) \frac{\rho^2}{2\alpha} \mathbb{1}\{|\mathbb{E}_\xi[G(x_{(m)}; \xi)]| \leq \rho\} \right], \\
&\leq \mathbb{E} \left[ \sum_{m=1}^{M_{RW}} \left\{ \frac{20\sigma_0^2}{\alpha} \log\left(\frac{2}{\check{p}} \log\left(\frac{40\sigma_0^2}{\check{p}\rho^2}\right)\right) + \frac{g_{\max}^2}{\alpha} \right\} \mathbb{1}\{|\mathbb{E}_\xi[G(x_{(m)}; \xi)]| > \rho\} \right. \\
&\quad \left. + \frac{20\sigma_0^2}{\alpha} \log\left(\frac{2}{\check{p}} \log\left(\frac{40\sigma_0^2}{\check{p}\rho^2}\right)\right) \mathbb{1}\{|\mathbb{E}_\xi[G(x_{(m)}; \xi)]| \leq \rho\} \right], \\
&\leq \left( \frac{20\sigma_0^2}{\alpha} \log\left(\frac{2}{\check{p}} \log\left(\frac{40\sigma_0^2}{\check{p}\rho^2}\right)\right) + \frac{g_{\max}^2}{\alpha} \right) \mathbb{E}[M_{RW}], \\
&\leq \left( \frac{20\sigma_0^2}{\alpha} \log\left(\frac{2}{\check{p}} \log\left(\frac{80\sigma_0^2}{\check{p}\alpha\epsilon}\right)\right) + \frac{g_{\max}^2}{\alpha} \right) \mathbb{E}[3N_{RW-\text{new}} + 3]. \tag{A.38}
\end{aligned}$$

Substituting the bound from Eqn. (A.35) in the above equation, we can show that for some constant  $\bar{R} > 0$ , independent of  $\epsilon$ ,

$$R_{RWT}(\epsilon) \leq \bar{R} \log\left(\frac{\mathbb{E}[f(x_0) - f(x^*)]}{\epsilon}\right) \log^2\left(\log\left(\frac{1}{\epsilon}\right)\right). \tag{A.39}$$

### A.3.4 Regret Analysis of PCM-RWT

We can now combine all the results obtained about performance of RWT to analyze the performance of PCM-RWT. Using the decomposition of regret in  $R_1$  and  $R_2$ , we bound each of these terms individually to obtain the bound on the overall regret.

We begin with bounding  $R_1$ . Note that we can now rewrite  $R_1$  as,

$$\begin{aligned}
R_1 &\leq \mathbb{E} \left[ \sum_{k=1}^K R_{RWT}(\epsilon_k) \right], \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K \bar{R} \log \left( \frac{\mathbb{E}[f(\mathbf{x}^{(k-1)}) - f(\mathbf{x}^*)]}{\epsilon_k} \right) \log^2 \left( \log \left( \frac{1}{\epsilon_k} \right) \right) \right], \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K \bar{R} \log \left( \frac{F_0 \gamma^k}{\epsilon_0 \gamma^k} \right) \log^2 \left( \log \left( \frac{1}{\epsilon_0} \right) + k \log \left( \frac{1}{\gamma} \right) \right) \right], \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K \bar{R}' \log^2 \left( \log \left( \frac{1}{\epsilon_0} \right) + k \log \left( \frac{1}{\gamma} \right) \right) \right]. \tag{A.40}
\end{aligned}$$

Using the result from Lemma 2.4.2 along with Jensen's inequality, we conclude that  $R_1$  is of the order  $O(\log T \log^2(\log T))$ . Similarly, we now consider  $R_2$ .

$$\begin{aligned}
R_2 &\leq \mathbb{E} \left[ \sum_{k=1}^K \sum_{t=t_{k-1}+1}^{t_k} [F(\mathbf{x}_{(i_k, \mathbf{x}^{(k-1)})}^*, \xi_t) - F(\mathbf{x}^*, \xi_t)] \right], \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K [f(\mathbf{x}^{(k-1)} - f(\mathbf{x}^*)) \mathbb{E}[\tau(\epsilon_k)]] \right], \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K (F_0 \gamma^{k-1}) \frac{\tau_0}{\epsilon_k} \log \left( \frac{\mathbb{E}[f(\mathbf{x}^{k-1}) - f(\mathbf{x}^*)]}{\epsilon_k} \right) \log^2 \left( \log \left( \frac{1}{\epsilon_k} \right) \right) \right], \\
&\leq \mathbb{E} \left[ \sum_{k=1}^K \tau'_0 \log^2 \left( \log \left( \frac{1}{\epsilon_0} \right) + k \log \left( \frac{1}{\gamma} \right) \right) \right]. \tag{A.41}
\end{aligned}$$

This is similar to the term we obtained in  $R_1$  implying that  $R_2$  is also of the order  $O(\log T \log^2(\log T))$ . Combining the two, we arrive at our required result.

## A.4 Parallel Implementation of PCM

The analysis of the parallel implementation scheme described in Section 2.6 is very similar to that of the sequential case. Let  $\mathbb{1}_{(i,j)}$  denote the indicator variable for the  $i^{\text{th}}$  direction and  $j^{\text{th}}$  core. It is 1 if the  $i^{\text{th}}$  direction was chosen for optimization on the  $j^{\text{th}}$  core where  $i = 1, 2, \dots, d$  and  $j = 1, 2, \dots, m$ . Thus, from Eqn. (A.17),

we have that for each  $j = 1, 2, \dots, m$ ,  $\mathbb{E}[f(\mathbf{y}_k)|\mathbf{x}] \leq f(\mathbf{x}) - \frac{1}{2\beta} \sum_{i=1}^d \mathbb{1}_{(i,j)}[g_i(\mathbf{x})]^2 + \epsilon$ , where  $\epsilon$  is the required accuracy. Using the update scheme in the synchronization step, we have,

$$\begin{aligned}
m\mathbb{E}[f(\mathbf{x}_1)|\mathbf{x}] &= m\mathbb{E}\left[f\left(\frac{1}{m} \sum_{j=1}^m \mathbf{y}_j\right) \middle| \mathbf{x}\right] \\
&\leq m\left(\frac{1}{m} \sum_{j=1}^m \mathbb{E}[f(\mathbf{y}_j)|\mathbf{x}]\right) \\
&\leq \sum_{j=1}^m \left(f(\mathbf{x}) - \frac{1}{2\beta} \sum_{i=1}^d \mathbb{1}_{(i,j)}[g_i(\mathbf{x})]^2 + \epsilon\right) \\
\implies \mathbb{E}[f(\mathbf{x}_1)|\mathbf{x}] &\leq \frac{1}{m} \left(\sum_{j=1}^m \left(f(\mathbf{x}) - \frac{1}{2\beta} \sum_{i=1}^d \mathbb{1}_{(i,j)}[g_i(\mathbf{x})]^2 + \epsilon\right)\right) \\
&\leq f(\mathbf{x}) - \frac{1}{2m\beta} \sum_{j=1}^m \sum_{i=1}^d \mathbb{1}_{(i,j)}[g_i(\mathbf{x})]^2 + \epsilon
\end{aligned}$$

where the second step follows from the convexity of the function. Now taking expectation over the random choice of coordinates, we get,

$$\begin{aligned}
\mathbb{E}[f(\mathbf{x}_1)|\mathbf{x}] &\leq f(\mathbf{x}) - \mathbb{E}\left[\frac{1}{2m\beta} \sum_{j=1}^m \sum_{i=1}^d \mathbb{1}_{(i,j)}[g_i(\mathbf{x})]^2\right] + \epsilon \\
&\leq f(\mathbf{x}) - \frac{1}{2m\beta} \sum_{j=1}^m \sum_{i=1}^d \frac{m}{d} [g_i(\mathbf{x})]^2 + \epsilon \\
&\leq f(\mathbf{x}) - \frac{m}{2d\beta} \sum_{i=1}^d [g_i(\mathbf{x})]^2 + \epsilon \\
&\leq f(\mathbf{x}) - \frac{m}{2d\beta} \|g(\mathbf{x})\|^2 + \epsilon \\
&\leq f(\mathbf{x}) - \frac{m\alpha}{d\beta} (f(\mathbf{x}) - f(\mathbf{x}^*)) + \epsilon
\end{aligned}$$

Note that this expression is similar to one obtained in Eqn. (A.17). Using an analysis similar to the one in Appendix A.1, we can obtain convergence rates

for the case of parallel updates. The reduction in dimensionality dependence is evident through the factor  $d$  being replaced by  $d/m$ .

APPENDIX B  
CHAPTER 3

We present some further details on the GP-ThreDS algorithm in Appendix B.1. Proof of Theorem 3.4.1 and all the lemmas are provided in Appendix B.2.

## B.1 Additional Details on GP-ThreDS Algorithm

In this section, we provide additional details on the implementation of GP-ThreDS.

### B.1.1 Simplified Implementation of Alg. 4

In the experiments, for a simpler implementation, we have devised the search for high-performing nodes at depth  $d$ , for a given input domain  $D$ , by directly searching among the leaf nodes. To be complete, a pseudo-code is given in Alg. 22 which is slightly different from the pseudo-code given in Alg. 4 regarding this step.

### B.1.2 Sequential test with asymmetric confidence levels

In this section, we describe the sequential test with asymmetric confidence levels. Recall the sequential test with symmetric confidence level  $\eta$ , given a node/region  $D$  and a threshold  $\tau$  described in Section 3.3.2 (see Fig. 3.3). In the

---

**Algorithm 22** Heuristic for Updating the domain

---

**Input:** Input domain  $D$ , discretization  $D_g$ ,  $C$ , confidence parameter  $\delta$ ,  $x_1 \in D_g$ , threshold  $\tau$

Set  $\text{HPNodes} \leftarrow \{\}$ ,  $\text{terminate} \leftarrow 0$ ,  $t \leftarrow 1$ ,  $t_{loc} \leftarrow 1$

**while**  $\text{terminate} \neq 1$  **do**

- if**  $\max_{x \in D_g} \mu_{t-1}(x) + \beta_t(\delta)\sigma_{t-1}(x) \leq \tau - L\Delta^\alpha$  **then**

  - $\text{terminate} \leftarrow 1$  // Stop the search

- else if**  $\max_{x \in D_g} \mu_{t-1}(x) - \beta_t(\delta)\sigma_{t-1}(x) \geq \tau$  **then**

  - $c^* \leftarrow \{c \in C : \arg \max_{x \in D_g} \mu_{t-1}(x) - \beta_t(\delta)\sigma_{t-1}(x) \in c\}$
  - $\text{HPNodes} \leftarrow \text{HPNodes} \cup c^*$  // Add the child node to the list of high-performing nodes
  - $C \leftarrow C \setminus c^*$  // Update the set of children
  - $D_g \leftarrow D_g \setminus \{x : x \in c^*\}$  // Update the set of search points
  - $t_{loc} \leftarrow 0$

- else if**  $t_{loc} == t_{term}$  **then**

  - $c^* \leftarrow \{c \in C : \arg \max_{x \in D_g} \mu_{t-1}(x) - \beta_t(\delta)\sigma_{t-1}(x) \in c\}$
  - $\text{HPNodes} \leftarrow \text{HPNodes} \cup c^*$
  - $C \leftarrow C \setminus c^*$
  - $D_g \leftarrow D_g \setminus \{x : x \in c^*\}$
  - $t_{loc} \leftarrow 0$

- end if**
- $x_t \leftarrow \arg \max_{x \in D_g} \mu_{t-1}(x) + \beta_t(\delta)\sigma_{t-1}(x)$
- Observe  $y_t = f(x_t) + \epsilon_t$
- Update  $t \leftarrow t + 1$ ,  $t_{loc} \leftarrow t_{loc} + 1$  and use the update equations to obtain  $\mu_t$  and  $\sigma_t$

**end while**

**return**  $\text{HPNodes}$

---

asymmetric case, similarly, we assume a threshold  $\tau$  and a node/region  $D$  is given. We consider two separate cases based on whether false positive rate is higher than false negative rate or vice versa.

In the first case, let  $1 - \hat{\delta}$  denote the confidence level for declaring whether node  $D$  is high-performing ( i.e., it contains a point with function value exceeding  $\tau$ ). Let  $1 - p$  denote the confidence level for declaring the node is not high-performing (i.e., it does not contain a point with function value exceeding  $\tau$ ). We assume that the false positive rate  $\hat{\delta}$  is lower than the false nega-

tive rate  $p$ . In this case the lower confidence bound and the upper confidence bound used in the test will initially be set to  $\max_{x \in D_g} \mu_{s-1}(x) - \beta_s(\hat{\delta})\sigma_{s-1}(x) \geq \tau$  and  $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(p)\sigma_{s-1}(x) \leq \tau - L\Delta^\alpha$ , respectively. After  $\bar{S}(p)$  steps, the upper confidence bound used in the test will be set to  $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\hat{\delta})\sigma_{s-1}(x) \leq \tau - L\Delta^\alpha$ . The number of samples is capped to  $\bar{S}(\hat{\delta})$ .

If the sequential test outputs a negative value, we have  $f(x_{D_g}^*) < \tau - L\Delta^\alpha$  with probability at least  $1 - p$ , (the probability will be higher, at least  $1 - \hat{\delta}$ , if the sequential test outputs a negative value after  $\bar{S}(p)$  steps). If the sequential test outputs a positive value, we have  $f(x_{D_g}^*) > \tau_k$  with a confidence of  $1 - \hat{\delta}$ . A pseudo code is provided in Alg. 23.

In the case where  $\hat{\delta}$  is the false negative rate and  $p$  is false positive rate ( $\hat{\delta} < p$ ), the upper and lower confidence bounds used in the test are set to  $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\hat{\delta})\sigma_{s-1}(x) \leq \tau - L\Delta^\alpha$  and  $\max_{x \in D_g} \mu_{s-1}(x) - \beta_s(p)\sigma_{s-1}(x) \geq \tau$ , respectively. The cap on the number of samples is set to  $S(\hat{\delta})$ .

In this case, if the sequential test outputs a positive value, we have  $f(x_{D_g}^*) > \tau_k$  with probability of at least  $1 - p$ . If the sequential test outputs a negative value, we have  $f(x_{D_g}^*) < \tau - L\Delta^\alpha$  with probability at least  $1 - \hat{\delta}$ . The cap on the number of samples is set to  $S(\hat{\delta})$  which helps us to focus our attention on the high confidence result of  $-1$  rather than on the low confidence result of  $+1$  resulting from the cap on the number of samples. The test may terminate (after  $\bar{S}(\hat{\delta})$  steps) and output a potentially erroneous positive value that is addressed in the analysis by the update rule of the thresholds  $\tau_k$  over epochs  $k$ . A pseudo code is provided in Alg. 24.

Lastly, while using this two-sided sequential test, the higher confidence level

is set to  $\hat{\delta}^{(r)} = \frac{\delta_0}{8Tr(r+1)(p-1/2)^2} \log\left(\frac{4dT}{\delta_0}\right)$  in the  $r^{\text{th}}$  run of the random walk procedure.

---

**Algorithm 23** Sequential Test with Asymmetric Confidence (low false positive rate)

---

**Input:** Discretization  $D_g$ , confidence parameters  $\hat{\delta}$  and  $p$ ,  $x_1 \in D_g$ , threshold  $\tau$   
 terminate  $\leftarrow 0$ ,  $s \leftarrow 1$

**while** terminate  $\neq 1$  **do**

**if**  $s < \bar{S}(p)$  **then**

**if**  $\max_{x \in D_g} \mu_{s-1}(x) - \beta_s(\hat{\delta})\sigma_{s-1}(x) \geq \tau$  **then**

terminate  $\leftarrow 1$ , retVal  $\leftarrow +1$

**else if**  $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(p)\sigma_{s-1}(x) \leq \tau - L\Delta^\alpha$  **then**

terminate  $\leftarrow 1$ , retVal  $\leftarrow -1$

**else**

Query  $x_s = \arg \max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\frac{\delta_0}{4T})\sigma_{s-1}(x)$

Observe  $y_s = f(x_s) + \epsilon_s$  and use it to obtain  $\mu_s$  and  $\sigma_s$

$s \leftarrow s + 1$

**end if**

**end if**

**if**  $s \geq \bar{S}(p)$  and  $s < \bar{S}(\hat{\delta})$  **then**

**if**  $\max_{x \in D_g} \mu_{s-1}(x) - \beta_s(\hat{\delta})\sigma_{s-1}(x) \geq \tau$  **then**

terminate  $\leftarrow 1$ , retVal  $\leftarrow +1$

**else if**  $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\hat{\delta})\sigma_{s-1}(x) \leq \tau - L\Delta^\alpha$  **then**

terminate  $\leftarrow 1$ , retVal  $\leftarrow -1$

**else**

Query  $x_s = \arg \max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\frac{\delta_0}{4T})\sigma_{s-1}(x)$

Observe  $y_s = f(x_s) + \epsilon_s$  and use it to obtain  $\mu_s$  and  $\sigma_s$

$s \leftarrow s + 1$

**end if**

**end if**

**if**  $s == \bar{S}(\hat{\delta})$  **then**

terminate  $\leftarrow 1$ , retVal  $\leftarrow +1$

**end if**

**end while**

**return** retVal

---

---

**Algorithm 24** Sequential Test with Asymmetric Confidence (low false negative rate)

---

```

Input: Discretization  $D_g$ , confidence parameters  $\hat{\delta}$  and  $p$ ,  $x_1 \in D_g$ , threshold  $\tau$ 
terminate  $\leftarrow 0$ ,  $s \leftarrow 1$ 
while terminate  $\neq 1$  do
    if  $s < \bar{S}(\hat{\delta})$  then
        if  $\max_{x \in D_g} \mu_{s-1}(x) - \beta_s(p)\sigma_{s-1}(x) \geq \tau$  then
            terminate  $\leftarrow 1$ , retVal  $\leftarrow +1$ 
        else if  $\max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\hat{\delta})\sigma_{s-1}(x) \leq \tau - L\Delta^\alpha$  then
            terminate  $\leftarrow 1$ , retVal  $\leftarrow -1$ 
        else
            Query  $x_s = \arg \max_{x \in D_g} \mu_{s-1}(x) + \beta_s(\frac{\delta_0}{4T})\sigma_{s-1}(x)$ 
            Observe  $y_s = f(x_s) + \epsilon_s$  and use it to obtain  $\mu_s$  and  $\sigma_s$ 
             $s \leftarrow s + 1$ 
        end if
    end if
    if  $s == \bar{S}(\hat{\delta})$  then
        terminate  $\leftarrow 1$ , retVal  $\leftarrow +1$ 
    end if
end while
return retVal

```

---

## B.2 Detailed Proofs

In this section, we provide the proof of Theorem 3.4.1, as well as all the lemmas that are used in the proof of theorems. We first state all the lemmas. We then provide the proof of Theorem 3.4.1 followed by the proof of the lemmas.

*Lemma 1.* For any set of sampling points  $\{x_1, x_2, \dots, x_t\}$  chosen from  $D_g$  (under any choice of algorithm), the following holds:  $\sum_{s=1}^t \sigma_{s-1}(x_s) \leq (1 + 2\lambda) \sqrt{|D_g|t}$ .

*Lemma 2.* If the local test is terminated by the termination condition at instant  $\bar{S}(\delta_2, \Delta_f)$  defined as  $\bar{S}(\delta_2, \Delta_f) = \min \left\{ t \in \mathbb{N} : 2(1 + 2\lambda)\beta_t(\delta_2) \sqrt{\frac{|D_g|}{t}} \leq \Delta_f \right\} + 1$ , then with probability at least  $1 - \delta_2$ , we have  $\tau - L\Delta^\alpha - \Delta_f \leq f(x_{D_g}^*) \leq \tau + \Delta_f$ .

*Lemma 3.* Consider the random walk based routine described in Section 3.3.2 with a local confidence parameter  $p \in (0, 1/2)$ . Then with probability at least

$1 - \delta_1$ , one iteration of RWT visits less than  $\frac{\log(d/\delta_1)}{2(p-1/2)^2}$  nodes before termination.

*Lemma 4.* The number of points in the discretization,  $|D_g|$ , for any node  $D$ , is upper bounded by a constant, independent of time. i.e.,  $|D_g| = O(1), \forall t \leq T$ .

**Lemma B.2.1.** *Consider the local test module carried out on a domain  $D$ , with a threshold  $\tau$  and a confidence parameter  $p \in (0, 1/2)$ , during an epoch  $k \geq 1$  of GP-ThreDS. If  $D$  contains a  $\tau$ -exceeding point, then the local test module outputs +1 with probability at least  $1 - p$ . If the local test outputs -1, then with probability at least  $1 - p$ ,  $D$  does not contain a  $\tau$ -exceeding point.*

**Lemma B.2.2.** *Let the interval in which  $f(x^*)$  lies, as maintained by the algorithm at the beginning of epoch  $k$ , be denoted by  $[a_k, b_k]$ . Then  $|b_k - a_k| \leq (1 + 2c(\rho_k/d - 1))2^{-\alpha(\rho_k/d - 1)}$ .*

### B.2.1 Proof of Theorem 3.4.1

For the regret analysis of GP-ThreDS, we write the overall regret as a sum of two terms,  $R_1$  and  $R_2$ .  $R_1$  is the regret incurred by the algorithm until the end of the epoch  $k_0$ , and  $R_2$  is the regret incurred by the algorithm after  $k_0$  epochs are completed, where  $k_0 = \max\{k : \rho_k \leq \frac{d}{2\alpha} \log T\}$ . All the following regret calculations are conditioned on the event that throughout the time horizon, all the random walk modules identify all the target nodes always correctly. We later show that this event occurs with a high probability.

We begin with the analysis of  $R_1$ . To obtain an upper bound on  $R_1$ , we first obtain the regret incurred at each node and sum that over the different nodes visited by the algorithm in the first  $k_0$  epochs. Since the sampling of the algorithm is independent across different nodes, we can bound the regret incurred

at any node  $D$  visited by the algorithm during an epoch  $k \leq k_0$  independent of others. We denote the discretized version of the domain by  $D_g$  and  $x_D^*$  and  $x_{D_g}^*$  are defined as follows:  $x_D^* = \arg \max_{x \in D} f(x)$  and  $x_{D_g}^* = \arg \max_{x \in D_g} f(x)$ . Recall that the cap on the number of samples in epoch  $k$  is defined as

$$\bar{S}^{(k)}(p) = \min \left\{ t \in \mathbb{N} : 2 \left( B + R \sqrt{2(\gamma_{t-1} + 1 + \log(1/p))} \right) (1 + 2\lambda) \sqrt{\frac{|D_g|}{t}} \leq L\Delta_k^\alpha \right\} + 1.$$

We focus our attention on any arbitrary node visited during the  $k^{\text{th}}$  epoch. Let  $N$  denote the random number of queries issued at that node and  $\bar{R}(N)$  denote the regret incurred at that node. By the definition of regret, we have,

$$\begin{aligned} \bar{R}(N) &= \sum_{n=1}^N f(x^*) - f(x_n) \\ &= \sum_{n=1}^N f(x^*) - \tau_k + L\Delta_k^\alpha + \tau_k - f(x_{D_g}^*) - L\Delta_k^\alpha + f(x_{D_g}^*) - f(x_n) \\ &= \underbrace{\left[ \sum_{n=1}^N f(x^*) - \tau_k + L\Delta_k^\alpha \right]}_{R^{(1)}(N)} + \underbrace{\left[ \sum_{n=1}^N \tau_k - L\Delta_k^\alpha - f(x_{D_g}^*) \right]}_{R^{(2)}(N)} + \underbrace{\left[ \sum_{n=1}^N f(x_{D_g}^*) - f(x_n) \right]}_{R^{(3)}(N)}. \end{aligned}$$

where  $x_n$  is the point sampled by the algorithm at the  $n^{\text{th}}$  time instant spent at the node. We will bound each of the three terms separately as outlined in Section 3.4.1.

We begin with bounding the third term,  $R^{(3)}(N)$ . Notice that it can be bounded in the same way as the regret for IGP-UCB since the sampling is always carried out on the grid by maximizing the UCB score over it. Since  $x_n = \arg \max_{x \in D_g} \mu_{n-1}(x) + \beta_n(\delta_0/4T)\sigma_{n-1}(x)$ , therefore, with probability at least  $1 - \delta_0/4T$ , we have

$$\begin{aligned} f(x_{D_g}^*) - f(x_n) &\leq \mu_{n-1}(x_{D_g}^*) + \beta_n(\delta_0/4T)\sigma_{n-1}(x_{D_g}^*) - (\mu_n(x_n) - \beta_n(\delta_0/4T)\sigma_{n-1}(x_n)) \\ &\leq \mu_{n-1}(x_n) + \beta_n(\delta_0/4T)\sigma_{n-1}(x_n) - \mu_n(x_n) + \beta_n(\delta_0/4T)\sigma_{n-1}(x_n) \\ &\leq 2\beta_n(\delta_0/4T)\sigma_{n-1}(x_n). \end{aligned}$$

From Lemma 3.4.2, we can conclude that  $\sum_{n=1}^N \sigma_{n-1}(x_n) \leq (1 + 2\lambda) \sqrt{|D_g| N}$ . Using this result along with the bound on  $f(x_{D_g}^*) - f(x_n)$ , we obtain

$$\begin{aligned} R^{(3)}(N) &= \sum_{n=1}^N f(x_{D_g}^*) - f(x_n) \\ &\leq \sum_{n=1}^N 2\beta_n(\delta_0/4T)\sigma_{n-1}(x_n) \\ &\leq 2\beta_N(\delta_0/4T) \sum_{n=1}^N \sigma_{n-1}(x_n) \\ &\leq 2(B + R \sqrt{2(\gamma_{N-1} + 1 + \log(4T/\delta_0))})(1 + 2\lambda) \sqrt{|D_g| N}. \end{aligned}$$

To bound the first term,  $R^{(1)}(N)$ , we relate the maximum number of samples taken at the node to  $f(x^*) - \tau_k + L\Delta_k^\alpha$  using Lemmas 3.4.3 and B.2.2. Recall the definition of  $\bar{S}^{(k)}(p)$ . It is defined as

$$\bar{S}^{(k)}(p) = \min \left\{ t \in \mathbb{N} : 2(B + R \sqrt{2(\gamma_{t-1} + 1 + \log(1/p))})(1 + 2\lambda) \sqrt{\frac{|D_g|}{t}} \leq L\Delta_k^\alpha \right\} + 1.$$

This implies that,

$$\begin{aligned} L\Delta_k^\alpha &\leq 2(B + R \sqrt{2(\gamma_{\bar{S}^{(k)}-3} + 1 + \log(1/p))})(1 + 2\lambda) \sqrt{\frac{|D_g|}{\bar{S}^{(k)} - 2}} \\ \implies 2^{-\alpha\rho_k/d} &\leq \frac{2}{c} (B + R \sqrt{2(\gamma_{\bar{S}^{(k)}} + 1 + \log(1/p))})(1 + 2\lambda) \sqrt{\frac{3|D_g|}{\bar{S}^{(k)}}} \end{aligned}$$

Notice that  $f(x^*)$  lies in  $[a_k, b_k]$  (under the high probability event on which the analysis is conditioned). Since  $\tau_k = (a_k + b_k)/2$ , therefore  $|f(x^*) - \tau_k| \leq |b_k - a_k|/2$ . Using this, we can write  $R^{(1)}(N)$  as,

$$\begin{aligned}
R^{(1)}(N) &= \sum_{n=1}^N f(x^*) - \tau_k + L\Delta_k^\alpha \\
&\leq (|f(x^*) - \tau_k| + L\Delta_k^\alpha) N \\
&\leq \left( \frac{|b_k - a_k|}{2} + c2^{-\alpha\rho_k/d} \right) N \\
&\leq \left( (1 + 2c(\rho_k/d - 1))2^{-\alpha(\rho_k/d-1)-1} + c2^{-\alpha\rho_k/d} \right) N \\
&\leq \left( (1 + 2c(\rho_k/d - 1))2^{-\alpha\rho_k/d} + c2^{-\alpha\rho_k/d} \right) N \\
&\leq \frac{2N}{c}(1 + c + 2c(\rho_k/d - 1)) \left( B + R \sqrt{2(\gamma_{\bar{S}^{(k)}} + 1 + \log(1/p))} \right) (1 + 2\lambda) \sqrt{\frac{3|D_g|}{\bar{S}^{(k)}}} \\
&\leq \frac{2N}{c}(1 + c + \frac{c}{\alpha} \log_2 T) \left( B + R \sqrt{2(\gamma_N + 1 + \log(1/p))} \right) (1 + 2\lambda) \sqrt{\frac{3|D_g|}{N}} \\
&\leq \frac{2}{c} \left( 2 + \frac{c}{\alpha} \log_2 T \right) \left( B + R \sqrt{2(\gamma_N + 1 + \log(1/p))} \right) (1 + 2\lambda) \sqrt{3|D_g|N},
\end{aligned}$$

where we use Lemma B.2.2 in line 4, definition of  $k_0$  in line 7 and the fact that  $N \leq \bar{S}$ . Lastly, we consider the second term,  $R^{(2)}(N)$ . Note that it is trivially upper bounded by zero if  $f(x_{D_g}^*) > \tau_k - L\Delta_k^\alpha$ . For the case when  $f(x_{D_g}^*) < \tau_k - L\Delta_k^\alpha$ , we analyze it like  $R^{(1)}(N)$  with a different time instant instead of  $\bar{S}^{(k)}(p)$ . Define  $t_1$  as

$$t_1 = \min \left\{ t \in \mathbb{N} : 2 \left( B + R \sqrt{2(\gamma_{t-1} + 1 + \log(4T/\delta_0))} \right) (1 + 2\lambda) \sqrt{\frac{|D_g|}{t}} \leq \tau_k - L\Delta_k^\alpha - f(x_{D_g}^*) \right\}.$$

From Lemma 3.4.3, we know that  $\Pr(N > t_1) \leq \frac{\delta_0}{4T}$ . Therefore, with probability at least  $1 - \frac{\delta_0}{4T}$ , we have  $N \leq t_1$ . Conditioning on this event and using a similar sequence of arguments as used in proof of  $R^{(1)}(N)$ , we can write

$$\tau_k - L\Delta_k^\alpha - f(x_{D_g}^*) \leq 2 \left( B + R \sqrt{2(\gamma_{t_1} + 1 + \log(4T/\delta_0))} \right) (1 + 2\sigma) \sqrt{\frac{2|D_g|}{t_1}}.$$

Thus with probability at least  $1 - \frac{\delta_0}{4T}$ , we have,

$$\begin{aligned}
R^{(2)}(N) &= \sum_{n=1}^N \tau_k - L\Delta_k^\alpha - f(x_{D_g}^*) \\
&\leq (\tau_k - L\Delta_k^\alpha - f(x_{D_g}^*)) N \\
&\leq 2N \left( B + R \sqrt{2(\gamma_{t_1} + 1 + \log(4T/\delta_0))} \right) (1 + 2\lambda) \sqrt{\frac{2|D_g|}{t_1}} \\
&\leq 2 \left( B + R \sqrt{2(\gamma_N + 1 + \log(4T/\delta_0))} \right) (1 + 2\lambda) \sqrt{2|D_g|N}.
\end{aligned}$$

On combining all the terms, we can conclude that  $\bar{R}(N)$  is  $O(\log T \sqrt{N(\gamma_N + \log(T/\delta_0))})$ .

To compute  $R_1$ , we just need to evaluate the total number of nodes visited by the algorithm in the first  $k_0$  epochs. Using Lemma 3.4.4, we can conclude that with probability at least  $1 - \delta_0/4T$ , one iteration of random walk would have visited less than  $\frac{1}{2(p-1/2)^2} \log\left(\frac{4dT}{\delta_0}\right)$  nodes. Therefore, throughout the algorithm, all iterations of random walks would have visited less than  $\frac{1}{2(p-1/2)^2} \log\left(\frac{4dT}{\delta_0}\right)$  nodes with probability at least  $1 - \delta_0/4$ .

Let  $L_m$  denote the number of nodes at depth  $md$  of tree  $\mathcal{T}_0$  for  $m = 1, 2, \dots, k_0$  that contain a point  $x$  such that  $f(x) \geq \tau_m - c2^{-\alpha\rho_m/d+1}$ . Therefore  $L_m$  denotes an upper bound on the number of target nodes for epoch  $m$ . Let  $L_0 = \max_{1 \leq i \leq k_0} L_i$ . Using the upper bound on the number of nodes visited during an iteration of RWT, we can conclude that the algorithm would have visited less than  $K = \frac{k_0 L_0}{(p-1/2)^2} \log\left(\frac{4dT}{\delta_0}\right)$  nodes in the first  $k_0$  epochs with probability at least  $1 - \delta_0/4$ . To bound  $k_0$ , note that the update scheme of the interval  $[a_k, b_k]$  (and consequently  $\tau_k$ ) ensures that the algorithm does not spend more than 2 epochs at any specific depth of the tree. This implies that  $k \leq 2\rho_k/d$ . Thus,  $k_0 \leq \frac{1}{\alpha} \log_2 T$ .

Let  $N_j$  denote the random number of queries at node  $j$  visited during the algorithm and  $\bar{R}_j(N_j)$  denote the associated regret for  $j = 1, 2, \dots, K$ . Therefore,

for some constant  $C_0$ , independent of  $T$ , we have,

$$\begin{aligned}
R_1 &\leq \sum_{j=1}^K \bar{R}_j(N_j) \\
&\leq C_0 \log T \sum_{j=1}^K \sqrt{N_j(\gamma_{N_j} + \log(T/\delta_0))} \\
&\leq C_0 \log T \sqrt{\gamma_T + \log(T/\delta_0)} \sum_{j=1}^K \sqrt{N_j} \\
&\leq C_0 \log T \sqrt{\gamma_T + \log(T/\delta_0)} \cdot \sqrt{K \sum_{j=1}^K N_j} \\
&\leq C_0 \log T \sqrt{\gamma_T + \log(T/\delta_0)} \cdot \sqrt{KT} \\
&\leq C_0 \sqrt{\frac{T \log(T)L_0}{2(p-1/2)^2} \log\left(\frac{4dT}{\delta_0}\right)} \cdot \sqrt{\gamma_T + \log(T/\delta_0)} \cdot \log T.
\end{aligned}$$

Therefore,  $R_1$  is  $O(\sqrt{T\gamma_T} \log T \sqrt{\log T \cdot \log(T/\delta_0)})$ .

We now focus on bounding  $R_2$ . Let  $D$  represent a node being visited after  $k_0$  epochs and  $x_D^* = \arg \max_{x \in D} f(x)$ . The instantaneous regret at time instant  $t$  can be written as

$$\begin{aligned}
r_t &= f(x^*) - f(x_t) \\
&= [f(x^*) - f(x_D^*)] + [f(x_D^*) - f(x_t)].
\end{aligned}$$

We bound both the expressions,  $f(x^*) - f(x_D^*)$  and  $f(x_D^*) - f(x_t)$  separately for any such node. We begin with the second expression. After  $k_0$  epochs, all the high-performing nodes being considered by the algorithm would be at a depth of at least  $\frac{d}{2\alpha} \log_2 T$  in the original infinite binary tree. This implies that the length of the edges of the cuboid corresponding to the nodes would be smaller than  $T^{-1/(2\alpha)}$ . Consequently, no two points in any such node would be more than  $\sqrt{d}T^{-1/(2\alpha)}$  apart. Therefore,  $f(x_D^*) - f(x_t) \leq L\sqrt{d^\alpha/T}$ , where  $x_t$  is a point sampled at time instant  $t$  after  $k_0$  epochs have been completed. To bound the first

expression, notice that  $f(x_D^*) \in [a_{k_0+1}, b_{k_0+1}]$  for all nodes visited after  $k_0$  epochs have been completed. This follows from the construction of intervals  $[a_k, b_k]$ . Since  $f(x^*)$  also lies in  $[a_{k_0+1}, b_{k_0+1}]$  (under the high probability event), we have,  $f(x^*) - f(x_D^*) \leq |b_{k_0+1} - a_{k_0+1}| \leq (1 + 2c(\rho_{k_0+1}/d - 1))2^{-\alpha(\rho_{k_0+1}/d-1)}$  for any node visited after  $k_0$  epochs. Therefore, we can bound the instantaneous regret as

$$\begin{aligned} r_t &= [f(x^*) - f(x_D^*)] + [f(x_D^*) - f(x_t)] \\ &\leq (1 + 2c(\rho_{k_0+1}/d - 1))2^{-\alpha(\rho_{k_0+1}/d-1)} + L \sqrt{\frac{d^\alpha}{T}} \\ &\leq 2(2 + \frac{c}{2\alpha} \log_2 T) \sqrt{\frac{1}{T}} + L \sqrt{\frac{d^\alpha}{T}}. \end{aligned}$$

If  $T_{R_2}$  denotes the samples taken by the algorithm after completing  $k_0$  epochs, then  $R_2$  can be bounded as

$$R_2 \leq \frac{T_{R_2}}{\sqrt{T}} \left( 2 \left( 2 + \frac{c}{2\alpha} \log_2 T \right) + L \sqrt{d^\alpha} \right).$$

Noting that  $T_{R_2} \leq T$ , we have that  $R_2$  is  $O(\sqrt{T} \log T)$ . On adding the bounds on  $R_1$  and  $R_2$ , we obtain that the regret incurred by the algorithm is  $O(\sqrt{T \gamma_T} \log T \sqrt{\log T \cdot \log(T/\delta_0)})$ , as required.

We now show that this bound holds with high probability. Firstly, we had obtained a bound on  $R^{(3)}(N)$  for a node  $D \subseteq \mathcal{X}$  by conditioning on the event that  $|f(x) - \mu_{t-1}(x)| \leq \beta_t(\delta_0/4T)\sigma_{t-1}(x)$  holds for all  $x \in D$  and  $t \geq 1$ . Since the probability that event occurs is at least  $1 - \delta_0/4T$ , the bound on  $R^{(3)}(N)$  holds simultaneously for all nodes visited throughout the time horizon with a probability of at least  $1 - \delta_0/4$ . Similarly, to obtain a bound on  $R^{(2)}(N)$  for any node  $D \subseteq \mathcal{X}$ , we had conditioned the analysis on another event which holds with a probability of at least  $1 - \delta_0/4T$ . Therefore, the bound on  $R^{(2)}(N)$  holds simultaneously for all nodes visited by the algorithm with a probability of at least  $1 - \delta_0/4$ . We also note that while using Lemma 3.4.4 to bound the number of nodes visited by the

algorithm, we had conditioned the analysis on another event (that bounded the number of nodes visited in an iteration of RWT) that holds with a probability of at least  $1 - \delta_0/4$  (See Appendix B.2.4). Lastly, since we assume that the algorithm always identifies all the target nodes correctly, we also need to account for the probability that this is true. From the error analysis of RWT as described in Appendix B.2.4, we note that every target node is identified correctly with a probability of at least  $1 - \delta_0/4T$ . Therefore, using a probability union bound, the algorithm identifies all the target nodes correctly with a probability of no less than  $1 - \delta_0/4$ . Combining all the above observations, we can conclude that the above obtained regret bound holds with a probability of at least  $1 - \delta_0$ , as required.

### B.2.2 Proof of Lemma 3.4.2

We consider a domain  $D \subseteq \mathcal{X}$  and its discretization  $D_g$  that contains  $|D_g|$  number of points. Let the points be indexed from 1 to  $|D_g|$  and let  $n_i$  denote the number of times the  $i^{\text{th}}$  point was chosen in the set of sampled points  $\{x_1, x_2, \dots, x_t\}$ . Let  $\mathcal{I} = \{i : n_i > 0\}$  and  $|\mathcal{I}|$  denote the number of elements in  $\mathcal{I}$ . Consider the  $i^{\text{th}}$  point, denoted by  $x^{(i)}$ , and let  $1 \leq t_1 < t_2 < \dots < t_{n_i} \leq t$  denote the time instances when the  $i^{\text{th}}$  point is sampled, that is, at time  $t_j$ , it is sampled for the  $j^{\text{th}}$  time, for  $j = 1, 2, \dots, n_i$ . Clearly, we have  $\sigma_{t_1-1}(x_{t_1}) = \sigma_{t_1-1}(x^{(i)}) \leq k(x^{(i)}, x^{(i)}) \leq 1$ . For all  $2 \leq j \leq n_i$ , at time instant  $t_j$ ,  $x^{(i)}$  has been sampled for  $j - 1$  times before  $t_j$ . Using Proposition 3 from Shekhar and Javidi [267], we have  $\sigma_{t_j-1}(x_{t_j}) = \sigma_{t_j-1}(x^{(i)}) \leq \frac{\lambda}{\sqrt{j-1}}$ . This can be interpreted as bounding the standard deviation by only the contribution coming from the noisy observations. We would like to emphasize that we are using the Proposition 3 for the *surrogate* GP-model adopted for the

optimization. While the actual noise is indeed  $R$ -sub-Gaussian, we are applying the Proposition 3 bearing in mind the *fictitious* Gaussian noise assumption for our *surrogate* model.

Thus for each point in  $\mathcal{I}$ , the contribution to the sum is upper bounded by  $1 + \lambda \sum_{j=1}^{n_i-1} j^{-1/2}$ . Thus, we have,

$$\begin{aligned} \sum_{s=1}^t \sigma_{s-1}(x_s) &\leq \sum_{i \in \mathcal{I}} \left( 1 + \lambda \sum_{j=1}^{n_i-1} \frac{1}{\sqrt{j}} \right) \\ &\leq \sum_{i \in \mathcal{I}} \left( 1 + \lambda \int_0^{n_i-1} \frac{1}{\sqrt{z}} dz \right) \\ &\leq \sum_{i \in \mathcal{I}} (1 + 2\lambda \sqrt{n_i - 1}) \\ &\leq (1 + 2\lambda) \sum_{i \in \mathcal{I}} \sqrt{n_i} \\ &\leq (1 + 2\lambda) |\mathcal{I}| \sqrt{\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} n_i} \\ &\leq (1 + 2\lambda) \sqrt{|\mathcal{I}| t}. \end{aligned}$$

In the fifth step, we have used Jensen's Inequality. Noting that  $|\mathcal{I}| \leq |D_g|$ , we obtain the required result.

### B.2.3 Proof of Lemma 3.4.3

Consider the performance of the local test on a domain  $D \subseteq \mathcal{X}$  with a threshold  $\tau$ . The discretized version of the domain is denoted by  $D_g$ . As before, we use the following notation throughout the proof of this lemma,. Let  $x_D^* = \arg \max_{x \in D} f(x)$ ,  $x_{D_g}^* = \arg \max_{x \in D_g} f(x)$ ,  $\hat{x}_t = \arg \max_{x \in D_g} \mu_{t-1}(x) + \beta_t(p)\sigma_{t-1}(x)$  and let  $\bar{x}_t = \arg \max_{x \in D_g} \mu_{t-1}(x) - \beta_t(p)\sigma_{t-1}(x)$ . Lastly, recall that the termination time is

defined as

$$\bar{S}(\delta_2, \Delta_f) = \min \left\{ t \in \mathbb{N} : 2 \left( B + R \sqrt{2(\gamma_{t-1} + 1 + \log(1/\delta_2))} \right) (1 + 2\lambda) \sqrt{\frac{|D_g|}{t}} \leq \Delta_f \right\} + 1.$$

Let us consider the case when  $f(x_{D_g}^*) < \tau - L\Delta^\alpha - \Delta_f$  and let  $N$  denote the random number of samples taken in a sequential test without a cap on the total number of samples. We first make the following observation about the posterior variance at the point to be sampled at  $t$ ,  $x_t$ , and  $\hat{x}_t$ . From the definitions of  $x_t$  and  $\hat{x}_t$ , we have,

$$\begin{aligned} \mu_{t-1}(x_t) + \beta_t(\delta_0/4T)\sigma_{t-1}(x_t) &\geq \mu_{t-1}(\hat{x}_t) + \beta_t(\delta_0/4T)\sigma_{t-1}(\hat{x}_t) \\ \mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) &\geq \mu_{t-1}(x_t) + \beta_t(p)\sigma_{t-1}(x_t) \end{aligned}$$

On adding the two, we obtain that  $\sigma_{t-1}(\hat{x}_t) \leq \sigma_{t-1}(x_t)$ . Note that this holds for all  $t$ . Next, we define the event  $E$  as  $|f(x) - \mu_{t-1}(x)| \leq \beta_t(\delta_2)\sigma_{t-1}(x)$  being true for all  $x \in D$  and  $t \geq 1$ . From [68, Theorem 2], we know that the probability of  $E$  is at least  $1 - \delta_2$ . Let  $E^c$  denote the complement of the event  $E$ . Using the event  $E$ , we evaluate the probability that the local test queries more than  $n$  points. The

probability that  $N > n$  can be written as follows,

$$\begin{aligned}
\Pr(N > n) &\leq \Pr(\{\forall t \leq n : \mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) \geq \tau - L\Delta^\alpha\}) \\
&\leq \Pr(\{\forall t \leq n : \mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) \geq \tau - L\Delta^\alpha\} | E) \Pr(E) + \\
&\quad \Pr(\{\forall t \leq n : \mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) \geq \tau - L\Delta^\alpha\} | E^c) \Pr(E^c) \\
&\leq \Pr\left(\sum_{t=1}^n \mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) \geq \sum_{t=1}^n (\tau - L\Delta^\alpha) \middle| E\right) + \Pr(E^c) \\
&\leq \Pr\left(\sum_{t=1}^n f(\hat{x}_t) + \beta_t(\delta_2)\sigma_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) \geq \sum_{t=1}^n (\tau - L\Delta^\alpha) \middle| E\right) + \delta_2 \\
&\leq \Pr\left(\sum_{t=1}^n f(x_{D_g}^*) + 2\beta_t(\delta_2)\sigma_{t-1}(\hat{x}_t) \geq \sum_{t=1}^n (\tau - L\Delta^\alpha) \middle| E\right) + \delta_2 \\
&\leq \Pr\left(\sum_{t=1}^n 2\beta_t(\delta_2)\sigma_{t-1}(x_t) \geq \sum_{t=1}^n (\tau - f(x_{D_g}^*) - L\Delta^\alpha) \middle| E\right) + \delta_2
\end{aligned}$$

To bound the first term on the RHS, we make use of Lemma 3.4.2.

Therefore, we have

$$\begin{aligned}
\frac{1}{n} \sum_{t=1}^n 2\beta_t(\delta_2)\sigma_{t-1}(x_t) &\leq \frac{2\beta_n(\delta_2)}{n} \sum_{t=1}^n \sigma_{t-1}(x_t) \\
&\leq \frac{2\beta_n(\delta_2)}{n} (1 + 2\lambda) \sqrt{|D_g|n} \\
&\leq 2\beta_n(\delta_2)(1 + 2\lambda) \sqrt{\frac{|D_g|}{n}} \\
&\leq \Delta_f < \tau - f(x_{D_g}^*) - L\Delta^\alpha.
\end{aligned}$$

This implies that the first term on RHS goes to zero for  $n \geq \bar{S} - 1$  implying that the probability that the local test takes more than  $\bar{S}$  samples when  $f(x_{D_g}^*) < \tau - L\Delta^\alpha - \Delta_f$  is less than  $\delta_2$ . This implies if the local test has reached the termination condition then with probability atleast  $1 - \delta_2$ , we have that  $f(x_{D_g}^*) > \tau - L\Delta^\alpha - \Delta_f$ .

We can carry out a similar analysis for the case when  $f(x_{D_g}^*) > \tau + \Delta_f$  to obtain the statement of the lemma.

### B.2.4 Proof of Lemma 3.4.4

The proof of this lemma is mainly based on the analysis of random walk on a binary tree. This analysis is similar to the one described in [313] (Also see Appendix A.3.2). We reproduce a slightly different version of the proof that is more focused on finding a high probability bound on the number of nodes visited in the random walk. In this proof, we consider a binary tree of depth  $d$ , denoted by  $\hat{\mathcal{T}}$ , to represent the tree considered in the random walk. We index the leaf nodes from 1 to  $n$  where  $n = 2^d$ . Throughout this proof, we refer to the high-performing nodes as target nodes. We begin with the case of a single target and then extend the proof for the case of multiple targets.

WLOG, we consider the single target node to be the leaf node indexed as

1. We divide the tree  $\hat{\mathcal{T}}$  into a sequence of sub-trees denoted by  $\hat{\mathcal{T}}_0, \hat{\mathcal{T}}_1, \dots$  for  $i = 0, 1, 2, \dots, d$  which are defined as follows. Consider the nodes on the path joining the root node to the target node. Such a path is unique as the underlying graph is a tree. Let  $v_i$  denote the node on this path that is at a distance of  $i$  from the target node. The distance between two nodes is defined as the length of the path connecting those two nodes.  $\hat{\mathcal{T}}_i$  is defined to be tree that contains the node  $v_i$  along with the sub-tree rooted at the child that does not contain the target node.

This construction is similar to the one outlined in [313]. Also,  $\hat{\mathcal{T}}_0$  corresponds to the target node. Since the random walk is biased towards the minimizer,

given the construction of  $\hat{\mathcal{T}}_i$ , the probability that random walk is still in one of such sub-trees would decrease with time. To formalize this idea, we consider the last passage times of any sub-tree  $\hat{\mathcal{T}}_i$  for  $1 \leq i \leq d$ . Let  $\tau_i$  denote the last passage time to  $\hat{\mathcal{T}}_i$ .

We begin with the analysis for  $\tau_d$ . This problem of random walk on  $\hat{\mathcal{T}}_d$  can be mapped to the problem of a random walk on the set  $S = \{-1, 0, 1, 2, \dots, d\}$ . If each non-negative integer is mapped to the subset of nodes at the corresponding depth in the sub-tree, then our random walk on  $\mathcal{T}_d$  between different levels is equivalent to the random walk on these integers. Note that since the target node is not contained in this sub-tree, all nodes at the same depth are identical in terms of distance to the target node. In particular, they all are equally far away from exiting the tree and therefore can be abstracted into single node. This abstraction is precisely what leads to the equivalence between the two problems. Under this setup, the root node is mapped to 0 and the sub-tree containing the target node is mapped to  $-1$ , indicating an exit from the sub-tree  $\hat{\mathcal{T}}_d$ .

We begin the random walk at integer 0 where escaping the tree is equivalent to arriving on the integer  $-1$ . For the random walk to arrive on  $-1$ , it would have to take greater number of steps in the negative direction than it took in the positive one. Also, since the probability of moving along the negative direction is at least  $1 - p$ , we can write,

$$\mathbb{P}(\tau_d > n) \leq \mathbb{P}(Z \leq n/2),$$

where  $Z \sim \text{Bin}(n, p)$  is a Binomial random variable. Therefore, we have

$$\mathbb{P}(\tau_d > n) \leq \exp(-2(p - 1/2)^2 n).$$

On account of the underlying symmetry, we can conclude that this bound holds for all  $i$ . Therefore, we have  $\mathbb{P}(\tau_i > n) \leq \exp(-2(p - 1/2)^2 n)$  for all  $i = 0, 1, \dots, d$ .

For the case of multiple target nodes, we can construct a similar set of sub-graphs and conclude the same result for those sub-graphs. Note that we redefine these set for every different iteration of the random walk when it restarts after detecting a target node. Consider the case when there are  $L$  target nodes. We begin with considering the first iteration of the random walk. For each target node  $j = 1, 2, \dots, L$ , we define a sequence of sub-trees  $\mathcal{T}_i^{(j)}$  for  $i = \{0, 1, \dots, d\}$  exactly in the same manner as we did in the previous case. That is,  $\mathcal{T}_i^{(j)}$  would be a tree consisting of the node that lies on the path between the target node  $j$  and the root node and is at a distance of  $i$  from the target node, along with child that does not contain the target node  $j$ . By definition, the sub-trees  $\mathcal{T}_i^{(j)}$  are not disjoint for different values of  $j$ . Using these sub-trees, we define a partition of the binary tree denoted by the sub-graphs  $\hat{\mathcal{T}}'_i$  for  $i = \{0, 1, \dots, d\}$  as follows. If  $\mathcal{V}$  denotes the set of all nodes on the binary tree, then for each  $v \in \mathcal{V}$ , we define  $v(j) = \{i : v \in \mathcal{T}_i^{(j)}\}$ . Therefore,  $v(j)$  denotes the index of the sub-tree corresponding to the target node  $j$  to which the node  $v$  belongs. From the construction of  $\mathcal{T}_i^{(j)}$ , it follows that  $v(j)$  is unique for each  $v \in \mathcal{V}$ . Using this, we define

$$\hat{\mathcal{T}}'_i = \{v \in \mathcal{V} : \min_j v(j) = i\}$$

In other words,  $\hat{\mathcal{T}}'_i$  consists of all the nodes such that there is at least one target node  $j$  for which it belongs to  $\mathcal{T}_i^{(j)}$ .

The motivation is that if the random walk escapes  $\hat{\mathcal{T}}'_i$  in the correct direction then it has moved closer to at least one of the target nodes. It is not difficult to note that this is exactly how the sub-trees  $\hat{\mathcal{T}}_i$  were designed in the previous proof. The only difference between the two cases is that  $\hat{\mathcal{T}}'_i$  is designed to accommodate the presence of multiple target nodes where all the target nodes have the same level of preference for the random walk. In a similar vein to the case of a single target, we define  $\tau'_i$  as the last passage time to  $\hat{\mathcal{T}}'_i$  for  $i = \{0, 1, \dots, d\}$ .

Leveraging the similarity of definitions of  $\hat{\tau}'_i$  and  $\hat{\tau}_i$  along with the agnosticism of the random walk to the target node, we can use exactly the same analysis as for the single target case to conclude that

$$\mathbb{P}(\tau'_i > n) \leq \exp(-2(p - 1/2)^2 n).$$

We let  $M$  denote the random number of steps taken by one iteration of random walk before termination. Therefore, we can write,

$$\begin{aligned} \Pr(M > r) &\leq \Pr\left(\bigcup_{i=0}^d \{\tau'_i > r\}\right) \\ &\leq \sum_{i=0}^d \Pr(\tau'_i > r) \\ &\leq \sum_{i=0}^d \exp(-2(p - 1/2)^2 r) \\ &\leq d \exp(-2(p - 1/2)^2 r) \end{aligned}$$

Using the above relation, we can conclude that one iteration of the random walk will take less than  $\frac{1}{2(p - 1/2)^2} \log\left(\frac{d}{\delta_1}\right)$  with probability at least  $1 - \delta_1$ , as required.

This also helps bound the probability of error in the random walk. If  $M_1$  denotes the number of non-target leaf nodes visited in the random walk, then the probability that a target node is identified incorrectly is less than  $M_1 \delta_2$ , where  $\delta_2$  is the error probability for the leaf test. Using the above bound on  $M_1$  with  $\delta_1 = \delta_0/4T$  along with the value of  $\delta_2$  as specified by the algorithm (See Section 3.3.2), we conclude that an iteration of random walk identifies a target node correctly with probability at least  $1 - \delta_0/4T$ .

### B.2.5 Proof of Lemma 3.4.7

A key idea in the proof of the lemma is to establish that for the choice of parameters used in GP-ThreDS, the rate at which domain shrinks matches the rate at which the discretization gets finer. Let  $D$  be a node visited by the algorithm during epoch  $k$  and  $D_g$  be its associated discretization such that

$$\sup_{x \in D} \inf_{y \in D_g} \|x - y\| \leq \Delta_k.$$

More specifically,  $D$  refers to the subset of the domain corresponding to the node visited by the algorithm.

Note that using the definition of a covering set, we can conclude that  $D_g$  is a  $\Delta_k$ -cover of  $D$ . Then, using the bounds on the covering number of a hypercube in  $\mathbb{R}^d$  [264], we have that  $|D_g|$  is  $O(\text{vol}(D)\Delta_k^{-d})$ . Since  $D$  is a node visited during epoch  $k$  of the algorithm, it lies at a depth of at least  $\rho_k - d$  on the infinite depth binary tree constructed on the domain. From the construction of the binary tree, we note that the lengths of nodes in all the dimensions get halved every  $d$  steps. Thus, the lengths of the edges of the cuboid corresponding to  $D$  are less than  $2^{-\rho_k/d+1}$ . Consequently,  $\text{vol}(D)$  is  $O(2^{-\rho_k})$ . On substituting this value in the bound for  $|D_g|$  along with  $\Delta_k = (c/L)^{1/\alpha} 2^{-\rho_k/d}$ , we obtain  $|D_g|$  is  $O(1)$ , independent of  $k$  (and thus  $t$ ). The exponential dependence of  $|D_g|$  on  $d$  also immediately follows from the above analysis.

As mentioned in Section 3.4.2, the proof of Theorem 3.4.6 follows from this lemma. Since only a constant number of UCB scores have to be evaluated at every time instant, the matrix inversion becomes the dominant cost resulting in a worst-case computational cost of  $O(t^3)$  at time  $t$ . Consequently, this results in worst-case overall computational complexity of  $O(T^4)$ .

### B.2.6 Proof of Lemma B.2.1

For the analysis of the local test, we consider several cases based on the maximum value of the function on the grid and consider the results obtained in each one of them.

We consider the performance of the local test on a node corresponding to  $D \subseteq \mathcal{X}$  visited by the random walk during epoch  $k$ . The discretized version of the domain is denoted by  $D_g$ . Recall that during epoch  $k$ , the closest point in  $D_g$  from any point  $x \in D$  is at a distance less than  $\Delta_k$ . We define  $x_D^*, x_{D_g}^*, \hat{x}_t$  and  $\bar{x}_t$  in the same way as in the proof of Lemma 3.4.3 (Appendix B.2.3).

The cap on the number of samples in epoch  $k$  is given as

$$\bar{S}^{(k)}(p) = \min \left\{ t \in \mathbb{N} : 2 \left( B + R \sqrt{2(\gamma_{t-1} + 1 + \log(1/p))} \right) (1 + 2\lambda) \sqrt{\frac{|D_g|}{t}} \leq L\Delta_k^\alpha \right\} + 1.$$

Similar to the proof of Lemma 3.4.3 (Appendix B.2.3), we define the event  $E$  as the inequality  $|f(x) - \mu_{t-1}(x)| \leq \beta_t(p)\sigma_{t-1}(x)$  being true for all  $x \in D$  and  $t \geq 1$ . We know this event occurs with a probability of at least  $1 - p$ . For the following analysis, we assume that event  $E$  occurs. Consider the following scenarios based on the value of  $f(x_{D_g}^*)$ .

- $f(x_{D_g}^*) > \tau_k + L\Delta_k^\alpha$ :

From the results obtained in the proof of Lemma 3.4.3, we know that the local test will not terminate. Also notice that the local test cannot return

-1 as

$$\begin{aligned}\mu_{t-1}(\hat{x}_t) + \beta_t(p)\sigma_{t-1}(\hat{x}_t) &\geq \mu_{t-1}(x_{D_g}^*) + \beta_t(p)\sigma_{t-1}(x_{D_g}^*) \\ &\geq f(x_{D_g}^*) \\ &> \tau_k - L\Delta_k^\alpha.\end{aligned}$$

Therefore, the local test will always return +1.

- $\tau_k + L\Delta_k^\alpha \geq f(x_{D_g}^*) \geq \tau_k$ :

Similar to the previous case, we can conclude that the local test will never return -1. It may return +1 or terminate.

- $\tau_k > f(x_{D_g}^*) > \tau_k - L\Delta_k^\alpha$ :

Again, similar to the previous cases, the local test will never return -1. For this case, we also have,

$$\begin{aligned}\mu_{t-1}(\bar{x}_t) - \beta_t(p)\sigma_{t-1}(\bar{x}_t) &\leq f(\bar{x}_t) \\ &\leq f(x_{D_g}^*) \\ &< \tau_k.\end{aligned}$$

Therefore, the local test will also never return +1 (before termination) implying it will always terminate.

- $\tau_k - L\Delta_k^\alpha \geq f(x_{D_g}^*) \geq \tau_k - 2L\Delta_k^\alpha$ :

Similarly, the local test will not return +1 (before termination). It may return -1 or terminate.

- $\tau_k - 2L\Delta_k^\alpha > f(x_{D_g}^*)$ :

From the results obtained in Section B.2.3 we can show the local will neither terminate nor return +1, implying that the local test will always return -1.

From the above analysis, one can directly obtain the statement of the lemma. If  $D$  is high-performing with respect to the threshold  $\tau_k$ , then  $f(x_D^*) > \tau_k$  implying that  $f(x_{D_g}^*) > \tau_k - L\Delta_k^\alpha$ . If  $f(x_{D_g}^*) > \tau_k - L\Delta_k^\alpha$ , then the local test will output  $+1$  whenever event  $E$  occurs, i.e., with a probability of at least  $1 - p$ . Similarly, if the local test outputs  $-1$  when  $E$  has occurred, we know that  $f(x_{D_g}^*) < \tau_k - L\Delta_k^\alpha$ , implying  $f(x_D^*) < \tau_k$  and hence  $D$  is not high-performing w.r.t. the threshold  $\tau_k$ , as required.

However, we would like to point out that when  $\tau_k - L\Delta_k^\alpha > f(x_{D_g}^*) > \tau_k - 2L\Delta_k^\alpha$ , the test may output  $-1$  or terminate, in which case it returns a  $+1$  and accept the current node. This happens because of the conservative nature of the local test. In order to avoid missing nodes that contain a point with a function value greater than  $\tau_k$ , the local test sometimes accepts nodes like these which have a point with a function value greater than  $\tau_k - c2^{-\alpha\rho_k/d+1}$  but not greater than  $\tau_k$ . This explains the reason behind the particular choice of values used in the update policy of  $\tau_k$ .

### B.2.7 Proof of Lemma B.2.2

We prove the statement of the lemma using induction. Recall that  $[a_k, b_k]$  denotes the interval to which  $f(x^*)$  is likely to belong at the beginning of epoch  $k$ . For the base case, for the LHS we have  $|b_1 - a_1| = |b - a| = 1$ . Since  $\rho_1 = d$ , the RHS also evaluates to 1 verifying the base case. Let us assume that the relation  $|b_k - a_k| \leq (1 + 2c(\rho_k - d)/d)2^{-\alpha(\rho_k/d-1)}$  is true for some  $k \geq 1$ .

In the event that the algorithm does not find any  $\tau_k$ -exceeding point, we set

$a_{k+1} = a_k - (b_k - a_k)/2$ ,  $b_{k+1} = b_k - (b_k - a_k)/2$  and  $\rho_{k+1} = \rho_k$ . Thus, we have,

$$\begin{aligned}|b_{k+1} - a_{k+1}| &= \left| b_k - \frac{b_k - a_k}{2} - a_k + \frac{b_k - a_k}{2} \right| \\&= |b_k - a_k| \\&\leq (1 + 2c(\rho_k - d)/d)2^{-\alpha(\rho_k/d-1)} \\&\leq (1 + 2c(\rho_{k+1} - d)/d)2^{-\alpha(\rho_{k+1}/d-1)},\end{aligned}$$

as required. Now, if the algorithm finds a  $\tau_k$ -exceeding point, we set  $a_{k+1} = \tau_k - c2^{-\alpha\rho_k/d+1}$ ,  $b_{k+1} = b_k$  and  $\rho_{k+1} = \rho_k + d$ . Thus, we have,

$$\begin{aligned}|b_{k+1} - a_{k+1}| &= |b_k - \tau_k + c2^{-\alpha\rho_k/d+1}| \\&\leq \frac{1}{2}|b_k - a_k| + c2^{-\alpha\rho_k/d+1} \\&\leq \frac{1}{2}\left(1 + \frac{2c(\rho_k - d)}{d}\right)2^{-\alpha(\rho_k/d-1)} + c2^{-\alpha\rho_k/d+1} \\&\leq \left(1 + \frac{2c(\rho_k - d)}{d}\right)2^{-\alpha\rho_k/d} + 2c2^{-\alpha\rho_k/d} \\&\leq \left(1 + 2c\left(\frac{(\rho_k + d) - d}{d}\right)\right)2^{-\alpha(\rho_{k+1}/d-1)} \\&\leq \left(1 + 2c\left(\frac{\rho_{k+1} - d}{d}\right)\right)2^{-\alpha(\rho_{k+1}/d-1)} \\&\leq \left(1 + \frac{2c(\rho_{k+1} - d)}{d}\right)2^{-\alpha(\rho_{k+1}/d-1)},\end{aligned}$$

as required. This completes the proof.

APPENDIX C  
CHAPTER 4

The appendix is structured as follows: In Appendix C.1, we provide the proof of Theorem 4.3.1, while we defer the proof of auxiliary lemmas to Appendix C.2. Proof of Theorem 4.3.3 is given in Appendix C.3. Further details on NeuralGCB and proof of Theorem 4.4.1 are provided in Appendix C.4.

## C.1 Proof of Theorem 4.3.1

Before we provide the proof of Theorem 4.3.1, we first set up some preliminaries and notation that would be useful throughout the proof.

### C.1.1 Proof Preliminaries

#### Notation

For any  $n \in \mathbb{N}$ ,  $[n]$  denotes the set  $\{1, 2, \dots, n\}$ . For any vector  $\mathbf{v} \in \mathbb{R}^n$ ,  $\text{diag}(\mathbf{v})$  is the  $\mathbb{R}^{n \times n}$  diagonal matrix with the elements on  $\mathbf{v}$  on its diagonal entries.  $\|\mathbf{v}\|$  denotes the  $L_2$  norm of the vector  $\mathbf{v}$ .  $\|\mathbf{M}\|_2$  and  $\|\mathbf{M}\|_F$  denotes the spectral and Frobenius norms respectively of a matrix  $\mathbf{M}$ . For events  $A$  and  $B$ , we define  $A \Rightarrow B := \neg A \vee B$ . For matrix  $\mathbf{A}$ , we denote the projection matrix for the column space of  $\mathbf{A}$  by  $\mathbf{\Pi}_A := \mathbf{A}\mathbf{A}^\dagger$ , where  $\mathbf{A}^\dagger$  denotes the pseudo-inverse of  $\mathbf{A}$ . Similarly, we denote the orthogonal projection matrix as  $\mathbf{\Pi}_A^\perp := \mathbf{I} - \mathbf{A}\mathbf{A}^\dagger$ . For two random variables,  $X$  and  $Y$ ,  $X \stackrel{d}{=} Y$  means  $X$  is equal to  $Y$  in distribution conditioned on the  $\sigma$ -algebra generated by the event  $A$ . For any  $\rho \in [-1, 1]$ , we use  $\Sigma_\rho$  to

denote the matrix  $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$ . Lastly, for  $n \in \mathbb{N}$ , we define  $(2n - 1)!! = \prod_{i=1}^n (2i - 1)$ . For example  $3!! = 3$  and  $5!! = 15$ .

## Fully connected Neural Network

In this proof, we consider a general fully connected neural net consisting of  $L$  hidden layers defined recursively as follows:

$$\mathbf{f}^{(l)}(\mathbf{x}) = \mathbf{W}^{(l)}\mathbf{h}^{(h-1)}(\mathbf{x}) \in \mathbb{R}^{d_l}, \quad \mathbf{h}^{(l)}(\mathbf{x}) = \sqrt{\frac{c_\sigma}{d_l}}\sigma(\mathbf{f}^{(l)}(\mathbf{x})) \in \mathbb{R}^{d_l}, \quad l = 1, 2, \dots, L, \quad (\text{C.1})$$

where  $\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x}$  and  $\mathbf{x} \in \mathcal{X}$  is the input to the network. In the above expression,  $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$  is the weight matrix in the  $l^{\text{th}}$  layer for  $l \in [L]$  and  $d_0 = d$ ,  $\sigma(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$  is a coordinate-wise activation function and the constant  $c_\sigma := (\mathbb{E}_{z \sim \mathcal{N}(0,1)}[\sigma(z)^2])^{-1}$ . The output of the neural network is given by its last layer defined as follows:

$$f(\mathbf{x}; \mathbf{W}) = \mathbf{f}^{(L+1)}(\mathbf{x}) = \mathbf{W}^{(L+1)}\mathbf{h}^{(L)}(\mathbf{x}), \quad (\text{C.2})$$

where  $\mathbf{W}^{(L+1)} \in \mathbb{R}^{1 \times d_L}$  is the weight matrix in the final layer, and  $\mathbf{W} = (\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L+1)})$  represents all the parameters of the network. Recall that the domain is assumed to be the hypersphere  $\mathbb{S}^{d-1}$ . Consequently,  $\|\mathbf{x}\| = 1$  for all  $\mathbf{x} \in \mathcal{X}$ . This is just the generalization of the of the neural net defined in Eqn. (4.2) with possibly different width in each layer.

The partial derivative of the output of the neural network with respect to a particular weight matrix is given as

$$\frac{\partial f(\mathbf{x}; \mathbf{W})}{\partial \mathbf{W}^{(l)}} = \mathbf{b}^{(l)}(\mathbf{x}) \cdot (\mathbf{h}^{(l-1)}(\mathbf{x}))^\top, \quad l = 1, 2, \dots, L + 1, \quad (\text{C.3})$$

where  $\mathbf{b}^{(l)}(\mathbf{x})$  is defined recursively as

$$\mathbf{b}^{(l)}(\mathbf{x}) = \begin{cases} 1 & l = L + 1, \\ \sqrt{\frac{c_\sigma}{d_l}} \mathbf{D}^{(l)}(\mathbf{x}) (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)}(\mathbf{x}) & l = 1, 2, \dots, L. \end{cases} \quad (\text{C.4})$$

In the above definition,

$$\mathbf{D}^{(l)}(\mathbf{x}) := \text{diag}(\sigma'(\mathbf{f}^{(l)}(\mathbf{x}))) \in \mathbb{R}^{d_l \times d_l}, \quad (\text{C.5})$$

is a diagonal matrix, where  $\sigma'(\cdot)$  is the derivative of the activation function  $\sigma$  and is also applied coordinate wise. Note that  $\mathbf{b}^{(l)}(\mathbf{x}) \in \mathbb{R}^{d_l}$  for  $l = 1, 2, \dots, L$  and  $\mathbf{b}^{(L+1)}(\mathbf{x}) \in \mathbb{R}$ . In other words,  $\mathbf{b}^{(l)}(\mathbf{x})$  is the gradient of output of the neural network  $f(\mathbf{x}; \mathbf{W})$  with respect to  $\mathbf{f}^{(l)}$ , the pre-activation of layer  $l$ .

### C.1.2 Neural Tangent Kernel

In the infinite width limit, the pre-activation functions  $\mathbf{f}^{(l)}$  at every hidden layer  $l \in [L]$  has all its coordinates tending to i.i.d. centered Gaussian Processes with covariance matrix  $\Sigma^{(l-1)} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$  defined recursively for  $l \in [L]$  as:

$$\begin{aligned} \Sigma^{(0)}(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^\top \mathbf{x}', \\ \Lambda^{(l)}(\mathbf{x}, \mathbf{x}') &= \begin{bmatrix} \Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}) & \Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}') \\ \Sigma^{(l-1)}(\mathbf{x}', \mathbf{x}) & \Sigma^{(l-1)}(\mathbf{x}', \mathbf{x}') \end{bmatrix}, \\ \Sigma^{(l)}(\mathbf{x}, \mathbf{x}') &= c_\sigma \mathbb{E}_{(u, v) \sim \mathcal{N}(0, \Lambda^{(l)}(\mathbf{x}, \mathbf{x}'))} [\sigma(u)\sigma(v)]. \end{aligned} \quad (\text{C.6})$$

Similar to  $\Sigma^{(l)}(\mathbf{x}, \mathbf{x}')$ , we also define  $\dot{\Sigma}^{(l)}(\mathbf{x}, \mathbf{x}')$  as follows:

$$\dot{\Sigma}^{(l)}(\mathbf{x}, \mathbf{x}') = c_\sigma \mathbb{E}_{(u, v) \sim \mathcal{N}(0, \Lambda^{(l)}(\mathbf{x}, \mathbf{x}'))} [\sigma'(u)\sigma'(v)], \quad (\text{C.7})$$

for  $l \in [L]$  and  $\dot{\Sigma}^{(L+1)}(\mathbf{x}, \mathbf{x}') = 1$  for  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ . The final NTK expression for the fully-connected network is given as

$$\Theta^{(L)}(\mathbf{x}, \mathbf{x}') = \sum_{l=1}^{L+1} \left( \Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}') \prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{x}, \mathbf{x}') \right). \quad (\text{C.8})$$

Note that this is same as  $k(\mathbf{x}, \mathbf{x}')$  referred to in the main text.

### C.1.3 The Activation Function

In this work, we assume that the activation function  $\sigma \in \mathcal{A}_\sigma$ , where  $\mathcal{A}_\sigma = \{\sigma_s(x) : s \in \mathbb{N}\}$  and  $\sigma_s(x) = (\max(0, x))^s$ . Note that  $\sigma_1(x)$  corresponds to the popular ReLU activation function and existing results hold only for  $\sigma_1(x)$ .

We state some definitions which we will be later used to establish certain properties of activation functions in  $\mathcal{A}$ .

**Fact 1.** ([300]) The normalizing constant  $c_{\sigma_s} = \frac{2}{(2s-1)!!}$  for all  $\sigma_s \in \mathcal{A}$ .

For simplicity of notation we use  $c_s$  instead of  $c_{\sigma_s}$  for the rest of the proof.

**Definition C.1.1.** A function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is said to be  $\alpha$ -homogeneous, if  $f(\lambda x) = \lambda^\alpha f(x)$  for all  $x \in \mathbb{R}$  and  $\lambda > 0$ .

It is straightforward to note that  $\sigma_s(x)$  is  $s$ -homogeneous.

Let  $\mathcal{M}_+(d)$  denote the set of positive semi-definite matrices of dimension  $d$ , that is,  $\mathcal{M}_+ = \{\mathbf{M} \in \mathbb{R}^{d \times d} : \mathbf{x}^\top \mathbf{M} \mathbf{x} \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^d\}$ . Similarly, we use  $\mathcal{M}_{++}(d)$  to denote the class of positive definite matrices of dimension  $d$ . For  $\gamma \in [0, 1]$ , we

denote

$$\mathcal{M}_+^\gamma = \left\{ \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{pmatrix} \in \mathcal{M}_+(2) \mid 1 - \gamma \leq \Sigma_{11}, \Sigma_{22} \leq 1 + \gamma \right\}. \quad (\text{C.9})$$

**Definition C.1.2.** The dual of an activation function  $\sigma(\cdot)$  is the function  $\bar{\sigma} : [-1, 1] \rightarrow \mathbb{R}$  defined as

$$\bar{\sigma}(\rho) = c_\sigma \mathbb{E}_{(X,Y) \sim \mathcal{N}(0, \Sigma_\rho)} [\sigma(X)\sigma(Y)]. \quad (\text{C.10})$$

Note that from definition of  $c_\sigma$ ,  $\bar{\sigma}(\rho) \in [-1, 1]$  and  $\bar{\sigma}(1) = 1$ .

**Fact 2** ([81, Lemma 11]). The dual  $\bar{\sigma}$  is continuous in  $[-1, 1]$ , smooth in  $(-1, 1)$ , convex in  $[0, 1]$  and is non-decreasing.

The dual of an activation function can be extended to  $\check{\sigma} : \mathcal{M}_+(2) \rightarrow \mathbb{R}$  as  $\check{\sigma}(\Sigma) = c_\sigma \mathbb{E}_{(X,Y) \sim \mathcal{N}(0, \Sigma)} [\sigma(X)\sigma(Y)]$ . Note that if  $\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{pmatrix}$  and  $\sigma$  is  $k$ -homogeneous, we have,

$$\check{\sigma}(\Sigma) = (\Sigma_{11}\Sigma_{22})^{k/2} \bar{\sigma}\left(\frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}}\right).$$

Let  $\bar{\sigma}_s$  be the dual function of  $\sigma_s \in \mathcal{A}_\sigma$  for all  $s \in \mathbb{N}$  and  $\bar{\mathcal{A}}_\sigma = \{\bar{\sigma}_s : s \in \mathbb{N}\}$  denote the set of dual functions. It is not difficult to note that  $\bar{\sigma}_s(-1) = 0$  and  $\bar{\sigma}_s(1) = 1$  for all  $s \in \mathbb{N}$ . Since  $\bar{\sigma}_s$  is non-decreasing,  $\bar{\sigma}_s(\rho) \in [0, 1]$  for all  $\rho \in [-1, 1]$ .

**Fact 3** ([300, Lemma 1]). The functions in  $\bar{\mathcal{A}}_\sigma$  satisfy,

$$\bar{\sigma}'_s(\rho) = \frac{s^2}{2s-1} \bar{\sigma}_{s-1}(\rho) \quad (\text{C.11})$$

for  $s > 1$ . Here  $\bar{\sigma}'_s$  denotes the derivative of  $\bar{\sigma}_s$ .

Consequently,  $|\bar{\sigma}'_s(\rho)| \leq \frac{s^2}{2s-1} |\bar{\sigma}_{s-1}(\rho)| \leq \frac{s^2}{2s-1}$ . Thus,  $\bar{\sigma}_s$  is  $s^2/(2s-1)$  Lipschitz.

**Definition C.1.3.** For any function  $\sigma_s \in \mathcal{A}_\sigma$ , we define  $\mu_{s,\rho} := \mathbb{E}_{(X,Y) \sim \mathcal{N}(0, \Sigma_\rho)}[\sigma_s(X)\sigma_s(Y)] = \frac{\bar{\sigma}_s}{c_s}$ .

### C.1.4 Proof of Lemma 4.3.2

The central piece in the proof of Theorem 4.3.1 is Lemma 4.3.2. We focus our attention on first proving Lemma 4.3.2. Since the proof is involved, we first provide an outline of the proof to give the reader an overview of the approach before delving into the technical details.

Informally, Lemma 4.3.2 states that for sufficiently large network widths, the following relation holds with probability of at least  $1 - \delta$  over the random initialization of the network weights.

$$\left| \left\langle \frac{\partial f(\mathbf{x}; \mathbf{W})}{\partial \mathbf{W}}, \frac{\partial f(\mathbf{x}'; \mathbf{W})}{\partial \mathbf{W}} \right\rangle - \Theta^{(L)}(\mathbf{x}, \mathbf{x}') \right| \leq (L+1)\varepsilon.$$

Firstly, note that

$$\left\langle \frac{\partial f(\mathbf{x}; \mathbf{W})}{\partial \mathbf{W}}, \frac{\partial f(\mathbf{x}'; \mathbf{W})}{\partial \mathbf{W}} \right\rangle = \sum_{l=1}^{L+1} \left\langle \frac{\partial f(\mathbf{x}; \mathbf{W})}{\partial \mathbf{W}^{(l)}}, \frac{\partial f(\mathbf{x}'; \mathbf{W})}{\partial \mathbf{W}^{(l)}} \right\rangle$$

and recall that

$$\Theta^{(L)}(\mathbf{x}, \mathbf{x}') = \sum_{l=1}^{L+1} \left( \Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}') \prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{x}, \mathbf{x}') \right).$$

Using these relations, note that it is sufficient to show that,

$$\left| \left\langle \frac{\partial f(\mathbf{x}; \mathbf{W})}{\partial \mathbf{W}^{(l)}}, \frac{\partial f(\mathbf{x}'; \mathbf{W})}{\partial \mathbf{W}^{(l)}} \right\rangle - \Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}') \prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{x}, \mathbf{x}') \right| \leq \varepsilon \quad (\text{C.12})$$

holds for all  $l \in [L]$  with probability  $1 - \delta$ . Furthermore, we have,

$$\begin{aligned} \left\langle \frac{\partial f(\mathbf{x}; \mathbf{W})}{\partial \mathbf{W}^{(l)}}, \frac{\partial f(\mathbf{x}'; \mathbf{W})}{\partial \mathbf{W}^{(l)}} \right\rangle &= \left\langle \mathbf{b}^{(l)}(\mathbf{x}) \cdot (\mathbf{h}^{(l-1)}(\mathbf{x}))^\top, \mathbf{b}^{(l)}(\mathbf{x}') \cdot (\mathbf{h}^{(l-1)}(\mathbf{x}'))^\top \right\rangle \\ &= \langle \mathbf{h}^{(l-1)}(\mathbf{x}), \mathbf{h}^{(l-1)}(\mathbf{x}') \rangle \langle \mathbf{b}^{(l)}(\mathbf{x}), \mathbf{b}^{(l)}(\mathbf{x}') \rangle. \end{aligned}$$

The proof revolves around establishing  $\langle \mathbf{h}^{(l-1)}(\mathbf{x}), \mathbf{h}^{(l-1)}(\mathbf{x}') \rangle$  is close to  $\Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}')$  while  $\langle \mathbf{b}^{(l)}(\mathbf{x}), \mathbf{b}^{(l)}(\mathbf{x}') \rangle$  is close to  $\prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{x}, \mathbf{x}')$ . On combining these two relations, we obtain the result in Eqn. (C.12) and consequently prove the theorem.

Throughout the proof, we fix some  $s \in \mathbb{N}$  and hence  $\sigma = \sigma_s$ . Recall from Eqn. (C.6), we have,

$$\begin{aligned}\Sigma^{(0)}(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^\top \mathbf{x}', \\ \Lambda^{(l)}(\mathbf{x}, \mathbf{x}') &= \begin{bmatrix} \Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}) & \Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}') \\ \Sigma^{(l-1)}(\mathbf{x}', \mathbf{x}) & \Sigma^{(l-1)}(\mathbf{x}', \mathbf{x}') \end{bmatrix}, \\ \Sigma^{(l)}(\mathbf{x}, \mathbf{x}') &= c_s \mathbb{E}_{(u, v) \sim \mathcal{N}(0, \Lambda^{(l)}(\mathbf{x}, \mathbf{x}'))} [\sigma_s(u)\sigma_s(v)].\end{aligned}$$

Since  $\|\mathbf{x}\| = 1$  for all  $x \in \mathcal{X}$ ,  $\Sigma^{(0)}(\mathbf{x}, \mathbf{x}) = 1$  for all  $x \in \mathcal{X}$ . Using induction, we can establish that  $\Sigma^{(l)}(\mathbf{x}, \mathbf{x}) = 1$  for all  $x \in \mathcal{X}$ , for all  $l \in \{0, 1, 2, \dots, L\}$ . The base case follows immediately as  $\Sigma^{(0)}(\mathbf{x}, \mathbf{x}) = 1$  for all  $x \in \mathcal{X}$ . Assume true for  $l - 1$ . Consequently, we have,  $\Lambda^{(l)}(\mathbf{x}, \mathbf{x}) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ . On plugging this value in the definition of  $\Sigma^{(l)}(\mathbf{x}, \mathbf{x})$ , we obtain  $\Sigma^{(l)}(\mathbf{x}, \mathbf{x}) = 1$ , completing the inductive step. As a result,  $|\Sigma^{(l)}(\mathbf{x}, \mathbf{x}')| \leq 1$  for all  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$  and hence we can write  $\Sigma^{(l)}(\mathbf{x}, \mathbf{x}') = \bar{\sigma}_s(\Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}'))$ .

As the final step before the proof, we define a sequence of events which will be used throughout the proof. Let  $\Delta^{(l)}(\mathbf{x}, \mathbf{x}') := \mathbf{D}^{(l)}(\mathbf{x})\mathbf{D}^{(l)}(\mathbf{x}')$ . We define the following events:

- $\mathcal{A}^l(\mathbf{x}, \mathbf{x}', \varepsilon_1) = \left\{ \left| \left( \mathbf{h}^{(l)}(\mathbf{x}) \right)^\top \mathbf{h}^{(l)}(\mathbf{x}') - \Sigma^{(l)}(\mathbf{x}, \mathbf{x}') \right| \leq \varepsilon_1 \right\},$
- $\bar{\mathcal{A}}^l(\mathbf{x}, \mathbf{x}', \varepsilon_1) = \mathcal{A}^l(\mathbf{x}, \mathbf{x}', \varepsilon_1) \cap \mathcal{A}^l(\mathbf{x}, \mathbf{x}, \varepsilon_1) \cap \mathcal{A}^l(\mathbf{x}', \mathbf{x}', \varepsilon_1)$
- $\bar{\mathcal{A}}(\mathbf{x}, \mathbf{x}', \varepsilon_1) = \bigcap_{l=0}^L \bar{\mathcal{A}}^l(\mathbf{x}, \mathbf{x}', \varepsilon_1)$

- $\mathcal{B}^l(\mathbf{x}, \mathbf{x}', \varepsilon_2) = \left\{ \left| \langle \mathbf{b}^{(l)}(\mathbf{x}), \mathbf{b}^{(l)}(\mathbf{x}') \rangle - \prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{x}, \mathbf{x}') \right| \leq \varepsilon_2 \right\}$
- $\bar{\mathcal{B}}^l(\mathbf{x}, \mathbf{x}', \varepsilon_2) = \mathcal{B}^l(\mathbf{x}, \mathbf{x}', \varepsilon_2) \cap \mathcal{B}^l(\mathbf{x}, \mathbf{x}, \varepsilon_2) \cap \mathcal{B}^l(\mathbf{x}', \mathbf{x}', \varepsilon_2)$
- $\bar{\mathcal{B}}(\mathbf{x}, \mathbf{x}', \varepsilon_2) = \bigcap_{l=1}^{L+1} \bar{\mathcal{B}}^l(\mathbf{x}, \mathbf{x}', \varepsilon_2)$
- $\bar{C}(\mathbf{x}, \mathbf{x}', \varepsilon_3) = \{|f(\mathbf{x}; \mathbf{W})| \leq \varepsilon_3, |f(\mathbf{x}'; \mathbf{W})| \leq \varepsilon_3\}$
- $\mathcal{D}^l(\mathbf{x}, \mathbf{x}', \varepsilon_4) = \left\{ c_s \frac{\text{trace}(\Delta^{(l)}(\mathbf{x}, \mathbf{x}'))}{d_l} - \dot{\Sigma}^{(l)}(\mathbf{x}, \mathbf{x}') \leq \varepsilon_4 \right\}$
- $\bar{\mathcal{D}}^l(\mathbf{x}, \mathbf{x}', \varepsilon_4) = \mathcal{D}^l(\mathbf{x}, \mathbf{x}', \varepsilon_4) \cap \mathcal{D}^l(\mathbf{x}, \mathbf{x}, \varepsilon_4) \cap \mathcal{D}^l(\mathbf{x}', \mathbf{x}', \varepsilon_4)$
- $\bar{\mathcal{D}}(\mathbf{x}, \mathbf{x}', \varepsilon_4) = \bigcap_{l=1}^{L+1} \bar{\mathcal{D}}^l(\mathbf{x}, \mathbf{x}', \varepsilon_4)$
- $\mathcal{E}^l(\mathbf{x}, \mathbf{x}', \varepsilon_5) = \{\|\Delta^{(l)}(\mathbf{x}, \mathbf{x}')\|_2 < \varepsilon_5\}$
- $\bar{\mathcal{E}}^l(\mathbf{x}, \mathbf{x}', \varepsilon_5) = \mathcal{E}^l(\mathbf{x}, \mathbf{x}', \varepsilon_5) \cap \mathcal{E}^l(\mathbf{x}, \mathbf{x}, \varepsilon_5) \cap \mathcal{E}^l(\mathbf{x}', \mathbf{x}', \varepsilon_5)$
- $\bar{\mathcal{E}}(\mathbf{x}, \mathbf{x}', \varepsilon_5) = \bigcap_{l=1}^{L+1} \bar{\mathcal{E}}^l(\mathbf{x}, \mathbf{x}', \varepsilon_5)$

We also state the following two Lemmas taken from Arora *et al.* [13] which are used at several points in the proof before beginning with the first part.

**Lemma C.1.4.** *For any two events, A and B,  $\Pr(A \Rightarrow B) \geq \Pr(B|A)$ .*

**Lemma C.1.5.** *Let  $\mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_d)$ ,  $\mathbf{G} \in \mathbb{R}^{d \times k}$  be a fixed matrix and  $\mathbf{F} = \mathbf{w}^\top \mathbf{G}$  be a random matrix. Then, conditioned on the value of  $\mathbf{F}$ ,  $\mathbf{w}$  remains Gaussian in the null space of the row space of  $\mathbf{G}$ . Mathematically,*

$$\Pi_G^\perp \mathbf{w} \stackrel{d}{=}_{\mathbf{F}=\mathbf{w}^\top \mathbf{G}} \Pi_G^\perp \tilde{\mathbf{w}},$$

where  $\tilde{\mathbf{w}}$  is a i.i.d. copy of  $\mathbf{w}$ .

## Pre-activations are close to the NTK matrix

In the first step, we show that  $\langle \mathbf{h}^{(l-1)}(\mathbf{x}), \mathbf{h}^{(l-1)}(\mathbf{x}') \rangle$  is close to  $\Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}')$ . We formalize this idea in the following lemma.

**Lemma C.1.6.** *For  $\sigma(z) = \sigma_s(z)$  and  $[\mathbf{W}^{(l)}]_{ij} \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 1)$  for all  $i \in [d_{l+1}], j \in [d_l]$  and  $l \in \{0, 1, 2, \dots, L\}$ . There exist constants  $c_1, c_2 > 0$  such that if  $\min_l d_l \geq c_1 \frac{s^L L^2 c_s^2}{\varepsilon^2} \log^2(sL/\delta\varepsilon)$  and  $\varepsilon \leq c_2/s$ , then for fixed  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ ,*

$$\left| \langle \mathbf{h}^{(l)}(\mathbf{y}), \mathbf{h}^{(l)}(\mathbf{y}') \rangle - \Sigma^{(l)}(\mathbf{y}, \mathbf{y}') \right| \leq \varepsilon,$$

holds with probability at least  $1 - \delta$  for all  $l \in \{0, 1, \dots, L\}$  and all  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$ . In other words,  $\Pr(\bar{\mathcal{A}}(\varepsilon)) \geq 1 - \delta$ , for fixed  $\mathbf{x}, \mathbf{x}'$ .

*Proof.* The proof of this Lemma relies on following lemmas.

**Lemma C.1.7.** *Let  $\rho \in [-1, 1]$  and  $(X, Y) \sim \mathcal{N}(0, \Sigma_\rho)$ . If  $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$  are independent samples from  $\mathcal{N}(0, \Sigma_\rho)$  and  $S_n = \frac{1}{n} \sum_{i=1}^n \sigma_s(X_i)\sigma_s(Y_i)$  for some  $\sigma_s \in \mathcal{A}$ , then for any  $t \geq 0$ ,*

$$\Pr(S_n - \mu_s \geq t) \leq \begin{cases} (n+1) \exp\left(-\frac{nt^2}{2\sqrt{3}\mu_{4s,\rho}}\right) & \text{if } t \leq t^*(n), \\ (n+1) \exp\left(-\frac{(nt)^{1/s}}{8(1+\rho)}\right) & \text{if } t > t^*(n). \end{cases}$$

$$\Pr(S_n - \mu_s \leq -t) \leq \exp\left(-\frac{nt^2}{2\mu_{2s,\rho}}\right),$$

where  $t^*(n) = \left(\frac{\sqrt{3}\mu_{4s,\rho}}{4(1+\rho)}\right)^{2-1/s} n^{-\left(\frac{s-1}{2s-1}\right)}$ . Consequently, if  $n \geq n_*(s, \delta, \rho)$ , then

$$\Pr\left(|S_n - \mu_{s,\rho}| \geq \sqrt{\frac{2(\sqrt{3}\mu_{4s,\rho} + \mu_{2s,\rho})}{n} \log\left(\frac{n+1}{\delta}\right)}\right) \leq \delta,$$

where  $n_*(s, \delta, \rho) := \min\left\{m \in \mathbb{N} : m \geq \frac{(8(1+\rho))^{2s}}{2\sqrt{3}\mu_{4s,\rho}} \left[\log\left(\frac{m+1}{\delta}\right)\right]^{2s-1}\right\} + 1$

For simplicity of notation, we define  $\varphi_s(n, \delta, \rho) := \sqrt{\frac{2(\sqrt{3\mu_{4,s,\rho}} + \mu_{2,s,\rho})}{n} \log\left(\frac{n+1}{\delta}\right)}$ .

Thus, the result of Lemma C.1.7 can be restated as  $\Pr(|S_n - \mu_{s,\rho}| \geq \varphi_s(n, \delta, \rho)) \leq \delta$ .

**Lemma C.1.8.** *For a given  $s \in \mathbb{N}$ , the dual activation function  $\check{\sigma}_s$  is  $\beta_s$ -Lipschitz in  $\mathcal{M}_+^{\gamma_s}$  w.r.t. the  $\infty$ -norm for  $\gamma_s \leq 1/s$  and  $\beta_s \leq 6s$ .*

**Lemma C.1.9** ([81]). *For  $\sigma(z) = \sigma_s(z)$  and  $[\mathbf{W}^{(l)}]_{ij} \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 1)$  for all  $i \in [d_{l+1}]$ ,  $j \in [d_l]$  and  $l \in \{0, 1, 2, \dots, L\}$ . If  $\min_{l \in [L]} d_l \geq n_s^*(\frac{\varepsilon}{B_{L,s}}, \frac{\delta}{8d})$ , then for fixed  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ ,*

$$\left| \langle \mathbf{h}^{(l)}(\mathbf{y}), \mathbf{h}^{(l)}(\mathbf{y}') \rangle - \Sigma^{(l)}(\mathbf{y}, \mathbf{y}') \right| \leq \varepsilon,$$

*holds with probability at least  $1 - \delta$  for all  $l \in \{0, 1, \dots, L\}$  and all  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$ . In the above expression,  $B_{L,s} = \sum_{j=0}^{L-1} \beta_{s'}^j \bar{d} = \sum_{l=1}^{L+1} d_l$  and  $n_s^*(\varepsilon, \delta) = \min\{n : \varphi_s(n, \delta) \leq \varepsilon\}$ .*

Lemma C.1.6 follows immediately from Lemma C.1.9 which in turn follows from the previous lemmas. The proofs of Lemma C.1.7 and C.1.8 are provided in Appendix C.2.

□

### Pre-activation gradients are close to NTK derivative matrices

In the second step, we show that  $\langle \mathbf{b}^{(l)}(\mathbf{x}), \mathbf{b}^{(l)}(\mathbf{x}') \rangle$  is close to  $\prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{x}, \mathbf{x}')$ . We formalize this idea in the following lemma.

**Lemma C.1.10.** *For  $\sigma = \sigma_s$  and  $[\mathbf{W}^{(l)}]_{ij} \stackrel{i.i.d.}{\sim} \mathcal{N}(0, 1)$  for all  $i \in [d_{l+1}]$ ,  $j \in [d_l]$  and  $l \in \{0, 1, 2, \dots, L\}$ . There exist constants  $c_1, c_2 > 0$  such that if  $\min_l d_l \geq c_1 \frac{s^L L^2 \varepsilon_s^2}{\varepsilon^2} \log^2(sL/\delta\varepsilon)$  and  $\varepsilon \leq c_2 s^{-L+2}/L$ , then for fixed  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ ,*

$$\left| \langle \mathbf{b}^{(l)}(\mathbf{y}), \mathbf{b}^{(l)}(\mathbf{y}') \rangle - \prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{y}, \mathbf{y}') \right| \leq L(\beta_s + 1)s^{L+1}\varepsilon,$$

holds with probability at least  $1 - \delta$  for all  $l \in \{0, 1, \dots, L\}$  and all  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$ . In other words, if  $\min_l d_l \geq c_1 \frac{s^L L^2 c_s^2}{\varepsilon_1^2} \log^2(sL/\delta\varepsilon_1)$  and  $\varepsilon_1 \leq c_2 s^{-L+2}/L$ , then for fixed  $\mathbf{x}, \mathbf{x}'$ ,  $\Pr(\bar{\mathcal{A}}(\varepsilon_1/2) \wedge \bar{\mathcal{B}}(L(\beta_s + 1)s^{L+1}\varepsilon_1)) \geq 1 - \delta$ .

*Proof.* We first state some helper lemmas that will be used in the proof followed by the proof of the above lemma.

**Lemma C.1.11** ([13, Lemma E.4]).

$$\Pr\left[\bar{\mathcal{A}}^L(\varepsilon/2) \Rightarrow \bar{C}\left(2\sqrt{\log\left(\frac{4}{\delta}\right)}\right)\right] \geq 1 - \delta \quad \forall \varepsilon \in [0, 1], \quad \forall \delta \in (0, 1).$$

**Lemma C.1.12.** If  $\bar{\mathcal{A}}^L(\varepsilon_1/2) \wedge \bar{\mathcal{B}}^{l+1}(\varepsilon_2) \wedge \bar{C}(\varepsilon_3) \wedge \bar{\mathcal{D}}^l(\varepsilon_4) \wedge \bar{\mathcal{E}}^l(\varepsilon_5)$  for any pair of points  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ , then for any  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$ , we have

$$\left| c_s \frac{\text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}'))}{d_l} \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle - \prod_{j=l}^L \dot{\Sigma}^{(l)}(\mathbf{y}, \mathbf{y}') \right| \leq s^{L-l}\varepsilon_4 + s\varepsilon_2.$$

**Lemma C.1.13.** If  $\bar{\mathcal{A}}^L(\varepsilon_1/2) \wedge \bar{\mathcal{B}}^{l+1}(\varepsilon_2) \wedge \bar{C}(\varepsilon_3) \wedge \bar{\mathcal{D}}^l(\varepsilon_4) \wedge \bar{\mathcal{E}}^l(\varepsilon_5)$  for any pair of points  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ , then for any  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$

$$\begin{aligned} & \left| \frac{c_s}{d_l} \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \boldsymbol{\Pi}_G^\perp \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \boldsymbol{\Pi}_G^\perp \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}) - \frac{c_s}{d_l} \text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}')) \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle \right| \\ & \leq c_s \left( \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}) \rangle + \langle \mathbf{b}^{(l+1)}(\mathbf{y}'), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle \right) \left( \sqrt{\frac{2}{d_l} \log\left(\frac{3}{\delta}\right)} + \frac{1}{d_l} \log\left(\frac{3}{\delta}\right) \right) \varepsilon_5 \\ & \quad + \frac{2c_s}{d_l} \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle \varepsilon_5. \end{aligned}$$

holds with probability at least  $1 - \delta$ . Consequently, for any  $\mathbf{y} \in \{\mathbf{x}, \mathbf{x}'\}$ ,

$$\sqrt{\frac{c_s}{d_l}} \left\| \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \boldsymbol{\Pi}_G^\perp \mathbf{D}^{(l)}(\mathbf{y}) \right\| \leq \sqrt{2c_s \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}) \rangle \varepsilon_5}.$$

**Lemma C.1.14.** If  $\bar{\mathcal{A}}^L(\varepsilon_1/2) \wedge \bar{\mathcal{B}}^{l+1}(\varepsilon_2) \wedge \bar{\mathcal{C}}(\varepsilon_3) \wedge \bar{\mathcal{D}}^l(\varepsilon_4) \wedge \bar{\mathcal{E}}^l(\varepsilon_5)$  for any pair of points  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ , then for any  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$

$$\begin{aligned} & \frac{c_s}{d_l} \left| \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}') \right. \\ & \quad \left. - \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \Pi_{\mathbf{G}}^\perp \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_{\mathbf{G}}^\perp \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}') \right| \leq \\ & \frac{c_s \varepsilon_5}{\sqrt{d_l}} \left[ (s^{(L-l)/2} + \varepsilon_2) \sqrt{2 \log \left( \frac{8}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] \times \\ & \quad \left\{ \frac{1}{\sqrt{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{8}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] + \sqrt{8}(s^{(L-l)/2} + 1) \right\} \end{aligned}$$

holds with probability at least  $1 - \delta$ .

**Lemma C.1.15.** For any  $\varepsilon_1 \in [0, 1/s]$ ,  $\varepsilon_2, \varepsilon_4 \in [0, 1]$  and  $\varepsilon_3, \varepsilon_5 \geq 0$ ,

$$\Pr \left[ \bar{\mathcal{A}}^L(\varepsilon_1/2) \wedge \bar{\mathcal{B}}^{l+1}(\varepsilon_2) \wedge \bar{\mathcal{C}}(\varepsilon_3) \wedge \bar{\mathcal{D}}^l(\varepsilon_4) \wedge \bar{\mathcal{E}}^l(\varepsilon_5) \Rightarrow \bar{\mathcal{B}}^l(\varepsilon') \right] \geq 1 - \delta,$$

where

$$\begin{aligned} \varepsilon' = & 2c_s(s^{L-l} + 1) \left( \sqrt{\frac{2}{d_l} \log \left( \frac{6}{\delta} \right)} + \frac{1}{d_l} \log \left( \frac{6}{\delta} \right) \right) \varepsilon_5 + \frac{2c_s}{d_l} (s^{L-l} + 1) \varepsilon_5 + s^{L-l} \varepsilon_4 + s \varepsilon_2 \\ & + \frac{c_s \varepsilon_5}{\sqrt{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{16}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] \times \\ & \quad \left\{ \frac{1}{\sqrt{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{16}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] + \sqrt{8}(s^{(L-l)/2} + 1) \right\}. \end{aligned}$$

We now prove Lemma C.1.10 by using induction on Lemma C.1.15. Before the inductive step, we note some relations that will be useful to us during the analysis. Firstly, using Lemma C.1.6, we can conclude that if  $d_l \geq n_s^*(\varepsilon/B_{L,s}, \delta/32\bar{d}(L+1))$  and  $\varepsilon \leq c_2/s$ , then

$$\Pr \left[ \bar{\mathcal{A}}^L(\varepsilon/2) \right] \geq 1 - \delta/4. \tag{C.13}$$

From Lemma C.1.11, we can conclude that

$$\Pr \left[ \bar{\mathcal{A}}^L(\varepsilon/2) \Rightarrow \bar{C} \left( 2 \sqrt{\log \left( \frac{16}{\delta} \right)} \right) \right] \geq 1 - \delta/4. \quad (\text{C.14})$$

If  $d_l \geq n_s^*(\varepsilon/3, \delta/24(L+1))$ , then we can conclude that  $3s^2\varphi_{s-1}((\min_l d_l), \delta/24(L+1)) + \frac{s^2\beta_s\varepsilon}{2s-1} \leq (\beta+1)s^2\varepsilon$ . Consequently for  $\varepsilon \leq 2/s$ ,

$$\Pr \left[ \bar{\mathcal{A}}(\varepsilon/2) \Rightarrow \bar{\mathcal{D}}((\beta_s+1)s^2\varepsilon) \wedge \bar{\mathcal{E}}(\varepsilon'') \right] \geq 1 - \delta/4, \quad (\text{C.15})$$

where  $\varepsilon'' = 3s^2 \left[ 2 \log \left( \frac{6(L+1)(\max_l d_l)}{\delta} \right) \right]^{s-1}$ . Applying the union bound on Eqn. (C.13), Eqn. (C.14) and Eqn. (C.15), we obtain that

$$\Pr \left[ \bar{\mathcal{A}}(\varepsilon/2) \wedge \bar{C} \left( 2 \sqrt{\log \left( \frac{16}{\delta} \right)} \right) \wedge \bar{\mathcal{D}}((\beta_s+1)s^2\varepsilon) \wedge \bar{\mathcal{E}}(\varepsilon'') \right] \geq 1 - 3\delta/4. \quad (\text{C.16})$$

Let  $d_l$  be chosen large enough to ensure that

$$\begin{aligned} s^{L-l+2}\varepsilon &\geq 2c_s(s^{L-l}+1) \left( \sqrt{\frac{2}{d_l} \log \left( \frac{24(L+1)}{\delta} \right)} + \frac{1}{d_l} \left( + \log \left( \frac{24(L+1)}{\delta} \right) \right) \right) \varepsilon'' \\ &+ c_s \varepsilon'' \sqrt{\frac{8}{d_l}} (s^{(L-l)/2}+1) \left[ (s^{(L-l)/2}+1) \sqrt{2 \log \left( \frac{64(L+1)}{\delta} \right)} + 4s^{L-l} \sqrt{\log \left( \frac{16(L+1)}{\delta} \right)} \right] \\ &+ \frac{c_s \varepsilon''}{d_l} \left[ (s^{(L-l)/2}+1) \sqrt{2 \log \left( \frac{64(L+1)}{\delta} \right)} + 4s^{L-l} \sqrt{\log \left( \frac{16(L+1)}{\delta} \right)} \right]^2. \end{aligned}$$

Define  $v_l := (L+1-l)(\beta_s+1)s^{L-l+2}\varepsilon$  for  $l = 0, 1, \dots, L+1$ . Invoking Lemma C.1.15 with  $\varepsilon_1 = \varepsilon$ ,  $\varepsilon_2 = v_{l+1}$ ,  $\varepsilon_3 = 2\sqrt{\log(\frac{16}{\delta})}$ ,  $\varepsilon_4 = (\beta_s+1)s^2\varepsilon$  and  $\varepsilon_5 = \varepsilon''$  gives us that

$$\Pr \left[ \bar{\mathcal{A}}^L(\varepsilon/2) \wedge \bar{\mathcal{B}}^{l+1}(v_{l+1}) \wedge \bar{C} \left( 2 \sqrt{\log \left( \frac{16}{\delta} \right)} \right) \wedge \bar{\mathcal{D}}^l((\beta_s+1)s^2\varepsilon) \wedge \bar{\mathcal{E}}^l(\varepsilon'') \Rightarrow \bar{\mathcal{B}}^l(v_l) \right] \geq 1 - \frac{\delta}{4(L+1)}.$$

Thus,

$$\begin{aligned}
& \Pr[\bar{\mathcal{A}}(\varepsilon/2) \wedge \bar{\mathcal{B}}(v_1)] \\
& \geq \Pr \left[ \underbrace{\bar{\mathcal{A}}(\varepsilon/2) \wedge \bar{C} \left( 2 \sqrt{\log \left( \frac{16}{\delta} \right)} \right) \wedge \bar{\mathcal{D}}((\beta_s + 1)s^2\varepsilon) \wedge \bar{\mathcal{E}}(\varepsilon'') \wedge \left( \bigcap_{l=1}^{L+1} \bar{\mathcal{B}}^l(v_l) \right)}_{:= \mathcal{F}(\varepsilon)} \right] \\
& \geq \Pr \left[ \mathcal{F}(\varepsilon) \wedge \bar{\mathcal{B}}^{L+1}(v_{L+1}) \wedge \bigcap_{l=0}^L \left\{ \mathcal{F}(\varepsilon) \wedge \left( \bigcap_{j=L+1-l}^{L+1} \bar{\mathcal{B}}^{j+1}(v_{j+1}) \right) \Rightarrow \bar{\mathcal{B}}^{L-l}(v_{L-l}) \right\} \right] \\
& \geq \Pr \left[ \mathcal{F}(\varepsilon) \wedge \bar{\mathcal{B}}^{L+1}(v_{L+1}) \wedge \bigcap_{l=0}^L \left\{ \bar{\mathcal{A}}^L(\varepsilon/2) \wedge \bar{C} \left( 2 \sqrt{\log \left( \frac{16}{\delta} \right)} \right) \wedge \bar{\mathcal{D}}^l((\beta_s + 1)s^2\varepsilon) \wedge \bar{\mathcal{E}}^l(\varepsilon'') \wedge \bar{\mathcal{B}}^{l+1}(v_{l+1}) \Rightarrow \bar{\mathcal{B}}^l(v_l) \right\} \right] \\
& \geq 1 - \Pr \left[ \neg \left\{ \bar{\mathcal{A}}(\varepsilon/2) \wedge \bar{C} \left( 2 \sqrt{\log \left( \frac{16}{\delta} \right)} \right) \wedge \bar{\mathcal{D}}((\beta_s + 1)s^2\varepsilon) \wedge \bar{\mathcal{E}}(\varepsilon'') \right\} \right] - \Pr \left[ \neg \bar{\mathcal{B}}^{L+1}(v_{L+1}) \right] \\
& \quad - \sum_{l=0}^L \Pr \left[ \neg \left\{ \bar{\mathcal{A}}^L(\varepsilon/2) \wedge \bar{C} \left( 2 \sqrt{\log \left( \frac{16}{\delta} \right)} \right) \wedge \bar{\mathcal{D}}^h((\beta_s + 1)s^2\varepsilon) \wedge \bar{\mathcal{E}}^l(\varepsilon'') \wedge \bar{\mathcal{B}}^{l+1}(v_{l+1}) \Rightarrow \bar{\mathcal{B}}^l(v_l) \right\} \right] \\
& \geq 1 - \frac{3\delta}{4} - \sum_{l=0}^L \frac{\delta}{4(L+1)} \\
& \geq 1 - \delta.
\end{aligned}$$

Thus,  $\Pr[\bar{\mathcal{A}}(\varepsilon/2) \wedge \bar{\mathcal{B}}(L(\beta_s + 1)s^{L+1}\varepsilon)] \geq 1 - \delta$ , as required. The proof of the helper lemmas are provided in Appendix C.2.

□

Now, it is straightforward to note that Lemma 4.3.2 immediately follows from Lemma C.1.6 and Lemma C.1.10. With Lemma 4.3.2 in place, the rest of the proof of Theorem 4.3.1 follows similar to the proof of Theorem 3.2 in [13]. The only step in the proof of Theorem 3.2 that is dependent on kernel being

ReLU, is Lemma F.6, where they bound the perturbation in gradient based on the amount of perturbation in the weight matrices. We would like to emphasize that this is only step apart from the ones which use Theorem 3.1. Such steps follow immediately by invoking Lemma 4.3.2 proven above, instead of Theorem 3.1. Using the result from [196, Theorem 3.2], we know that the spectral norm of the Hessian of the neural net is  $O(m^{-1/2})$ . Here the implied constant only depends on  $s$  and  $L$ . Thus for the a perturbation in the weight matrices of the order of  $O(\sqrt{m}\omega)$ , the corresponding perturbation in the gradient is  $O(\omega)$ , with the implied constant depending only on  $s$  and  $L$ . On substituting this relation in place of Lemma F.6, the claim of Theorem 4.3.1 follows using the same arguments as the ones used in the proof of Theorem 3.2 of Arora *et al.* [13].

## C.2 Proof of Helper Lemmas

We first state some additional intermediate lemmas which are used in the proofs several helper lemmas then provide a proof of all the lemmas.

**Definition C.2.1.**

$$\mathbf{G}^{(l)}(\mathbf{x}, \mathbf{x}') = [\mathbf{h}^{(l)}(\mathbf{x}) \ \mathbf{h}^{(l)}(\mathbf{x}')] \in \mathbb{R}^{d_l \times 2}$$

$$\tilde{\mathbf{G}}^{(l)}(\mathbf{x}, \mathbf{x}') = [\mathbf{G}^{(l)}(\mathbf{x}, \mathbf{x}')]^\top \mathbf{G}^{(l)}(\mathbf{x}, \mathbf{x}') = \begin{bmatrix} \langle \mathbf{h}^{(l)}(\mathbf{x}), \mathbf{h}^{(l)}(\mathbf{x}) \rangle & \langle \mathbf{h}^{(l)}(\mathbf{x}), \mathbf{h}^{(l)}(\mathbf{x}') \rangle \\ \langle \mathbf{h}^{(l)}(\mathbf{x}'), \mathbf{h}^{(l)}(\mathbf{x}) \rangle & \langle \mathbf{h}^{(l)}(\mathbf{x}'), \mathbf{h}^{(l)}(\mathbf{x}') \rangle \end{bmatrix} \equiv \begin{bmatrix} G_{11} & G_{12} \\ G_{12} & G_{22} \end{bmatrix}$$

**Lemma C.2.2.**

$$\Pr\left[\bar{\mathcal{A}}^l(\varepsilon) \Rightarrow \bar{\mathcal{D}}^{l+1}\left(3s^2\varphi_{s-1}(d_l, \delta/6, 1) + \frac{2s^2\beta_s\varepsilon}{2s-1}\right) \wedge \bar{\mathcal{E}}^{l+1}\left(3s^2\left[2\log\left(\frac{6d_l}{\delta}\right)\right]^{s-1}\right)\right] \geq 1 - \delta$$

$$\forall \varepsilon \in [0, 1/s], \delta \in (0, 1).$$

**Lemma C.2.3.** Let

$$\varepsilon' = 3s^2\varphi_{s-1}((\min_l d_l), \delta/6(L+1), 1) + \frac{2s^2\beta_s\varepsilon}{2s-1}, \quad \varepsilon'' = 3s^2 \left[ 2 \log \left( \frac{6(L+1)(\max_l d_l)}{\delta} \right) \right]^{s-1}.$$

Then,

$$\Pr \left[ \bar{\mathcal{A}}(\varepsilon) \Rightarrow \bar{\mathcal{D}}(\varepsilon') \wedge \bar{\mathcal{E}}(\varepsilon'') \right] \geq 1 - \delta \quad \forall \varepsilon \in [0, 1/s], \delta \in (0, 1).$$

**Lemma C.2.4.** For any  $1 \leq l \leq L$ , any fixed  $\{\mathbf{W}^{(i)}\}_{i=1}^{l-1}$  and  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ , with probability  $1 - \delta$  over the randomness of  $\mathbf{W}^{(l)}$ , we have,

$$\left| c_s \frac{\text{trace}(\Delta^{(l)}(\mathbf{x}, \mathbf{x}'))}{d_l} - \frac{s^2}{2s-1} \check{\sigma}_{s-1}(\tilde{\mathbf{G}}^{(l)}(\mathbf{x}, \mathbf{x}')) \right| \leq s^2 (G_{11} G_{22})^{\frac{(s-1)}{2}} \varphi_{s-1}(d_l, \delta/2, \rho_G),$$

and

$$\|\Delta^{(l)}(\mathbf{x}, \mathbf{x}')\|_2 \leq s^2 \left[ (\sqrt{G_{11} G_{22}} + G_{12}) \log \left( \frac{2d_l}{\delta} \right) \right]^{s-1},$$

where  $\rho_G = \frac{G_{12}}{\sqrt{G_{11} G_{22}}}$ .

**Lemma C.2.5** ([39]). Let  $\xi \sim \mathcal{N}(0, \mathbf{I}_n)$  be an  $n$ -dimensional unit Gaussian random vector and  $\mathbf{A} \in \mathbb{R}^{n \times n}$  be a symmetric matrix, then for any  $t > 0$ ,

$$\Pr \left( |\xi^\top \mathbf{A} \xi - \mathbb{E}[\xi^\top \mathbf{A} \xi]| > 2\|\mathbf{A}\|_F \sqrt{t} + 2\|\mathbf{A}\|_2 t \right) \leq \exp(-t),$$

or equivalently,

$$\Pr \left( |\xi^\top \mathbf{A} \xi - \mathbb{E}[\xi^\top \mathbf{A} \xi]| > t \right) \leq \exp \left( -\frac{t^2}{4\|\mathbf{A}\|_F^2 + 4\|\mathbf{A}\|_2} \right).$$

**Lemma C.2.6.** If  $\bar{\mathcal{A}}^L(\varepsilon_1/2) \wedge \bar{\mathcal{B}}^{l+1}(\varepsilon_2) \wedge \bar{\mathcal{C}}(\varepsilon_3) \wedge \bar{\mathcal{D}}^l(\varepsilon_4) \wedge \bar{\mathcal{E}}^l(\varepsilon_5)$  for any pair of points  $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ , then for any  $\mathbf{y} \in \{\mathbf{x}, \mathbf{x}'\}$

$$\left\| \mathbf{\Pi}_G \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}) \right\| \leq (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{4}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2}.$$

holds with probability at least  $1 - \delta$ .

### C.2.1 Proof of Lemma C.1.7

We fix a particular  $s \in \mathbb{N}$ . Let  $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$  be  $n$  independent samples from  $\mathcal{N}(0, \Sigma_\rho)$  and let  $Z_i = \sigma_s(X_i)\sigma_s(Y_i)$ . Define  $S_n := \frac{1}{n} \sum_{i=1}^n Z_i$ . Note that  $\mathbb{E}[S_n] = \mathbb{E}[Z_1] = \mu_s$ . To prove the bound, we first bound  $\Pr(Z_i > t)$  for any  $t \geq 0$  and then using techniques borrowed from Gantert *et al.* [112], we obtain an upper bound on  $\Pr(S_n - \mu_s > t)$ . The bound on  $\Pr(S_n - \mu_s < -t)$  follows from Cramer's Theorem [75].

We begin with bounding  $\Pr(Z_i > t)$  for any  $t \geq 0$ . Since  $Z_i = (\sigma_1(X_i)\sigma_1(Y_i))^s$ , we just focus on bounding  $\Pr(\sigma_1(X)\sigma_1(Y) > t)$  for  $(X, Y) \sim \mathcal{N}(0, \Sigma_\rho)$ . Fix a  $t \geq 0$ . We have,

$$\begin{aligned}\Pr(\sigma_1(X)\sigma_1(Y) > t) &= \mathbb{E} [\mathbb{1}_{\{\sigma_1(X)\sigma_1(Y) > t\}}] \\ &= \mathbb{E} [\mathbb{1}_{\{XY > t, X \geq 0, Y \geq 0\}}].\end{aligned}$$

Using a change of variables define  $U = (X + Y)/\sqrt{2}$  and  $V = (X - Y)/\sqrt{2}$ . Note that  $U$  and  $V$  are zero-mean Gaussian random variables with  $\text{Cov}(U, V) = 0$  implying that they are independent. Also  $\text{Var}(U) = 1 + \rho$  and  $\text{Var}(V) = 1 - \rho$ .

Thus,

$$\begin{aligned}
\Pr(\sigma_1(X)\sigma_1(Y) > t) &= \mathbb{E}_{X,Y} [\mathbb{1}\{XY > t, X \geq 0, Y \geq 0\}] \\
&= \mathbb{E}_{U,V} [\mathbb{1}\{U^2 - V^2 > 2t, U \geq 0, V \in [-U, U]\}] \\
&\leq \mathbb{E}_{U,V} [\mathbb{1}\{U^2 > V^2 + 2t, U \geq 0\}] \\
&\leq \mathbb{E}_V [\Pr(U > \sqrt{V^2 + 2t})] \\
&\leq \mathbb{E}_V \left[ \exp\left(-\frac{V^2 + 2t}{2(1+\rho)}\right) \right] \\
&\leq \frac{1}{\sqrt{2\pi(1-\rho)}} \int_{-\infty}^{\infty} \exp\left(-\frac{v^2 + 2t}{2(1+\rho)}\right) \exp\left(-\frac{v^2}{2(1-\rho)}\right) dv \\
&\leq \frac{e^{-t/(1+\rho)}}{\sqrt{2\pi(1-\rho)}} \int_{-\infty}^{\infty} \exp\left(-\frac{v^2}{(1-\rho^2)}\right) dv \\
&\leq \exp\left(-\frac{t}{1+\rho}\right) \sqrt{\frac{1+\rho}{2}} \\
&\leq \exp\left(-\frac{t}{1+\rho}\right).
\end{aligned}$$

In the sixth line we used the concentration bound for Gaussian random variable and in the eighth line we used the Gaussian integral. Consequently,

$$\Pr(Z_i > t) = \Pr(\sigma_1(X)\sigma_1(Y) > t^{1/s}) \leq \exp\left(-\frac{t^{1/s}}{1+\rho}\right). \quad (\text{C.17})$$

Using this relation, we now move on to bound  $\Pr(S_n \geq x)$  for  $x \geq \mu_{s,\rho}$ . For the rest of the proof we drop the argument  $\rho$  for simplicity. Firstly, note that

$$\Pr(S_n \geq x) \leq \underbrace{\Pr\left(\max_{j \in [n]} Z_j \geq n(x - \mu_s)\right)}_{A_1^n} + \underbrace{\Pr\left(S_n \geq x, \max_{j \in [n]} Z_j < n(x - \mu_s)\right)}_{A_2^n}.$$

We begin with the first term.

$$\Pr\left(\max_{j \in [n]} Z_j \geq n(x - \mu_s)\right) \leq n \Pr(Z_i \geq n(x - \mu_s)) \leq n \exp\left(-\frac{n^{1/s}(x - \mu_s)^{1/s}}{1+\rho}\right).$$

Let  $\beta_\zeta(n) = \frac{\zeta n^{1/s}}{1+\rho} > 0$  for some  $\zeta > 0$  which will be specified later. We have,

$$\begin{aligned}
A_2^n &= \Pr\left(S_n \geq x, \max_{j \in [n]} Z_j < n(x - \mu_s)\right) \\
&= \Pr\left(\exp(\beta_\zeta(n)S_n) \geq \exp(\beta_\zeta(n)x), \max_{j \in [n]} Z_j < n(x - \mu_s)\right) \\
&\leq \exp(-\beta_\zeta(n)x) \mathbb{E}\left[\exp(\beta_\zeta(n)S_n) \mathbb{1}_{\left\{\max_{j \in [n]} Z_j < n(x - \mu_s)\right\}}\right] \\
&\leq \exp(-\beta_\zeta(n)x) \prod_{j=1}^n \mathbb{E}\left[\exp\left(\frac{\beta_\zeta(n)Z_j}{n}\right) \mathbb{1}_{\{Z_j < n(x - \mu_s)\}}\right] \\
\implies \log(A_2^n) &\leq -\beta_\zeta(n)x + \sum_{j=1}^n \Gamma_\zeta^{(j)}(n),
\end{aligned}$$

where  $\Gamma_\zeta^{(j)}(n) = \log\left(\mathbb{E}\left[\exp\left(\frac{\beta_\zeta(n)Z_j^{(n)}}{n}\right)\right]\right)$  and  $Z_j^{(n)} := Z_j \mathbb{1}_{\{Z_j < n(x - \mu_s)\}}$ . Using the relations  $\log(1+x) \leq x$  and  $e^x \leq 1+x+\frac{x^2}{2}e^x$ , we have,

$$\begin{aligned}
\sum_{j=1}^n \Gamma_\zeta^{(j)}(n) &= \sum_{j=1}^n \log\left(\mathbb{E}\left[\exp\left(\frac{\beta_\zeta(n)Z_j^{(n)}}{n}\right)\right]\right) \\
&\leq \sum_{j=1}^n \log\left(1 + \mathbb{E}\left[\frac{\beta_\zeta(n)Z_j^{(n)}}{n}\right] + \frac{1}{2}\mathbb{E}\left[\left(\frac{\beta_\zeta(n)Z_j^{(n)}}{n}\right)^2 \exp\left(\frac{\beta_\zeta(n)Z_j^{(n)}}{n}\right)\right]\right) \\
&\leq \sum_{j=1}^n \left\{ \mathbb{E}\left[\frac{\beta_\zeta(n)Z_j^{(n)}}{n}\right] + \frac{1}{2}\mathbb{E}\left[\left(\frac{\beta_\zeta(n)Z_j^{(n)}}{n}\right)^2 \exp\left(\frac{\beta_\zeta(n)Z_j^{(n)}}{n}\right)\right] \right\}.
\end{aligned}$$

Note that,

$$\begin{aligned}
\sum_{j=1}^n \mathbb{E}\left[\frac{\beta_\zeta(n)Z_j^{(n)}}{n}\right] &= \sum_{j=1}^n \frac{\beta_\zeta(n)}{n} \mathbb{E}[Z_j^{(n)}] \\
&\leq \beta_\zeta(n) \mu_s.
\end{aligned}$$

For the second term we have,

$$\mathbb{E}\left[\left(\frac{\beta_\zeta(n)Z_1^{(n)}}{n}\right)^2 \exp\left(\frac{\beta_\zeta(n)Z_1^{(n)}}{n}\right)\right] \leq \left(\frac{\beta_\zeta(n)}{n}\right)^2 \mathbb{E}\left[\left(Z_1^{(n)}\right)^{\frac{2(1+\eta)}{\eta}}\right]^{\frac{\eta}{1+\eta}} \mathbb{E}\left[\exp\left(\frac{(1+\eta)\beta_\zeta(n)Z_1^{(n)}}{n}\right)\right]^{\frac{1}{1+\eta}}$$

for some  $\eta > 0$  by using Hölder's inequality. Since  $\mathbb{E}\left[\left(Z_1^{(n)}\right)^{2(1+\eta)/\eta}\right]^{\eta/(1+\eta)} =$

$\left[\mu_{\frac{2s(1+\eta)}{\eta}}\right]^{\frac{\eta}{1+\eta}}$ , we focus on the other term. We have,

$$\begin{aligned}
& \mathbb{E}\left[\exp\left(\frac{(1+\eta)\beta_\zeta(n)Z_1^{(n)}}{n}\right)\right] \\
&= 1 + \int_0^{n(x-\mu_s)} \frac{(1+\eta)\beta_\zeta(n)}{n} \exp\left(\frac{(1+\eta)\beta_\zeta(n)z}{n}\right) \Pr(Z_1 > z) dz \\
&= 1 + n(x-\mu_s) \int_0^1 \frac{(1+\eta)\beta_\zeta(n)}{n} \exp\left(\frac{(1+\eta)\beta_\zeta(n)n(x-\mu_s)y}{n}\right) \Pr(Z_1 > n(x-\mu_s)y) dy \\
&\leq 1 + (x-\mu_s) \int_0^1 (1+\eta)\beta_\zeta(n) \exp\left((1+\eta)\beta_\zeta(n)(x-\mu_s)y - \frac{(n(x-\mu_s)y)^{1/s}}{1+\rho}\right) dy \\
&\leq 1 + (x-\mu_s)(1+\eta)\beta_\zeta(n) \int_0^1 \exp\left(\beta_\zeta(n)\left[(1+\eta)(x-\mu_s)y - \frac{(x-\mu_s)^{1/s}y^{1/s}}{\zeta}\right]\right) dy.
\end{aligned}$$

For  $\zeta \leq \frac{(x-\mu_s)^{1/s-1}}{2(1+\eta)}$ , we have,

$$\begin{aligned}
& \mathbb{E}\left[\exp\left(\frac{(1+\eta)\beta_\zeta(n)Z_1^{(n)}}{n}\right)\right] \\
&\leq 1 + (x-\mu_s)(1+\eta)\beta_\zeta(n) \int_0^1 \exp\left(\beta_\zeta(n)\left[(1+\eta)(x-\mu_s)y - \frac{(x-\mu_s)^{1/s}y^{1/s}}{\zeta}\right]\right) dy \\
&\leq 1 + (x-\mu_s)(1+\eta)\beta_\zeta(n) \int_0^1 \exp\left(-\frac{1}{2}\beta_\zeta(n)(1+\eta)(x-\mu_s)y\right) dy \\
&\leq 3.
\end{aligned}$$

On combining all the equations, we obtain that for  $\zeta \leq \frac{(x-\mu_s)^{1/s-1}}{2(1+\eta)}$  and  $\eta > 0$

$$\log(A_2^n) \leq -\beta_\zeta(n)(x-\mu_s) + \frac{(\beta_\zeta(n))^2}{2n} \left(\left[\mu_{\frac{2s(1+\eta)}{\eta}}\right]^{\frac{\eta}{1+\eta}} 3^{\frac{1}{1+\eta}}\right)$$

We set  $\eta = 1$ . Thus for  $\zeta \leq 0.25(x-\mu_s)^{1/s-1}$ , we have,

$$\log(A_2^n) \leq -\frac{\zeta n^{1/s}(x-\mu_s)}{1+\rho} + \frac{\zeta^2 n^{2/s-1}}{2(1+\rho)^2} \sqrt{3\mu_{4s}}.$$

Note that the RHS is minimized at  $\zeta_* = \frac{n^{1-1/s}(x-\mu_s)(1+\rho)}{\sqrt{3\mu_{4s}}}$ . However,  $\zeta_* \leq 0.25(x-\mu_s)^{1/s-1}$  only for  $(x-\mu_s) \leq \underbrace{\left(\frac{\sqrt{3\mu_{4s}}}{4(1+\rho)}\right)^{2-1/s} n^{-\left(\frac{s-1}{2s-1}\right)}}_{:=t^*(n)}$ . Thus, for  $(x-\mu_s) \leq t^*(n)$ ,

we can plug in  $\zeta = \zeta_*$  in the above expression to obtain

$$\log(A_2^n) \leq -\frac{n(x - \mu_s)^2}{2\sqrt{3\mu_{4s}}}.$$

For  $(x - \mu_s) > t^*(n)$ , we plug in  $\zeta = 0.25(x - \mu_s)^{1/s-1}$  to obtain

$$\log(A_2^n) \leq -\frac{n^{1/s}(x - \mu_s)^{1/s}}{8(1 + \rho)}.$$

On combining the two, we obtain that for any  $t \geq 0$

$$\Pr(S_n - \mu_s \geq t) = \begin{cases} (n+1) \exp\left(-\frac{nt^2}{2\sqrt{3\mu_{4s}}}\right) & \text{if } t \leq t^*(n), \\ (n+1) \exp\left(-\frac{(nt)^{1/s}}{8(1+\rho)}\right) & \text{if } t > t^*(n). \end{cases}$$

We now consider the other side. Consider any  $x \leq \mu_s$  and  $\lambda > 0$ . We have,

$$\begin{aligned} \Pr(S_n \leq x) &= \Pr(\exp(-\lambda S_n) \geq e^{-\lambda x}) \\ &\leq \mathbb{E}[\exp(-\lambda S_n)] e^{\lambda x} \\ &\leq \mathbb{E}\left[1 - \lambda S_n + \frac{\lambda^2 S_n^2}{2}\right] e^{\lambda x} \\ &\leq \left(1 - \lambda \mathbb{E}[S_n] + \frac{\lambda^2 \mathbb{E}[S_n^2]}{2}\right) e^{\lambda x} \\ &\leq \left(1 - \lambda \mu_s + \frac{\lambda^2 \mu_{2s}}{2n}\right) e^{\lambda x} \\ &\leq \exp\left(-\lambda \mu_s + \frac{\lambda^2 \mu_{2s}}{2n} + \lambda x\right) \\ &\leq \exp\left(\lambda(x - \mu_s) + \frac{\lambda^2 \mu_{2s}}{2n} + \lambda x\right). \end{aligned}$$

Minimizing this over  $\lambda$  yields  $\lambda_* = -\frac{n(x - \mu_s)}{\mu_{2s}} > 0$ . On plugging this value, we obtain,

$$\Pr(S_n \leq x) \leq \exp\left(-\frac{n(x - \mu_s)^2}{2\mu_{2s}}\right).$$

Consequently for any  $t \geq 0$ , we have,

$$\Pr(S_n - \mu_s \leq -t) \leq \exp\left(-\frac{nt^2}{2\mu_{2s}}\right).$$

On combining the relations, we can conclude that if  $n \geq n_*(s, \delta, \rho)$ , then

$$\Pr \left( |S_n - \mu_s| \geq \sqrt{\frac{2(\sqrt{3\mu_{4s}} + \mu_{2s})}{n} \log \left( \frac{n+1}{\delta} \right)} \right) \leq \delta.$$

### C.2.2 Proof of Lemma C.1.8

Fix a  $s \in \mathbb{N}$ . We drop the subscript  $s$  for the remainder of the proof for ease of notation. We need to show that the dual activation function  $\check{\sigma}$  is  $\beta$ -Lipschitz in  $\mathcal{M}_+^\gamma$  w.r.t. the  $\infty$ -norm. Let  $\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{pmatrix} \in \mathcal{M}_+^\gamma$ . In order to prove  $\beta$ -Lipschitzness w.r.t. the  $\infty$ -norm, it is sufficient to show that,

$$\|\nabla \check{\sigma}_s\|_1 = \left| \frac{\partial \check{\sigma}_s}{\partial \Sigma_{12}} \right| + \left| \frac{\partial \check{\sigma}_s}{\partial \Sigma_{11}} \right| + \left| \frac{\partial \check{\sigma}_s}{\partial \Sigma_{22}} \right| \leq \beta.$$

Recall that since  $\sigma_s$  is  $s$ -homogeneous,

$$\check{\sigma}_s(\Sigma) = (\Sigma_{11}\Sigma_{22})^{s/2} \bar{\sigma}_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right).$$

Using this relation, we have,

$$\begin{aligned} \frac{\partial \check{\sigma}_s}{\partial \Sigma_{12}} &= (\Sigma_{11}\Sigma_{22})^{\frac{s-1}{2}} \bar{\sigma}'_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) \\ \frac{\partial \check{\sigma}_s}{\partial \Sigma_{11}} &= \frac{s}{2} (\Sigma_{11}\Sigma_{22})^{\frac{s}{2}-1} \Sigma_{22} \bar{\sigma}_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) - \frac{1}{2} (\Sigma_{11}\Sigma_{22})^{\frac{s-1}{2}} \bar{\sigma}'_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) \frac{\Sigma_{12}}{\Sigma_{11}} \\ &= \Sigma_{22} \left\{ \frac{s}{2} (\Sigma_{11}\Sigma_{22})^{\frac{s}{2}-1} \bar{\sigma}_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) - \frac{1}{2} (\Sigma_{11}\Sigma_{22})^{\frac{s-1}{2}} \bar{\sigma}'_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) \frac{\Sigma_{12}}{\Sigma_{11}\Sigma_{22}} \right\} \\ \frac{\partial \check{\sigma}_s}{\partial \Sigma_{22}} &= \frac{s}{2} (\Sigma_{11}\Sigma_{22})^{\frac{s}{2}-1} \Sigma_{11} \bar{\sigma}_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) - \frac{1}{2} (\Sigma_{11}\Sigma_{22})^{\frac{s-1}{2}} \bar{\sigma}'_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) \frac{\Sigma_{12}}{\Sigma_{22}} \\ &= \Sigma_{11} \left\{ \frac{s}{2} (\Sigma_{11}\Sigma_{22})^{\frac{s}{2}-1} \bar{\sigma}_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) - \frac{1}{2} (\Sigma_{11}\Sigma_{22})^{\frac{s-1}{2}} \bar{\sigma}'_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) \frac{\Sigma_{12}}{\Sigma_{11}\Sigma_{22}} \right\}. \end{aligned}$$

Consequently,

$$\begin{aligned} \|\nabla \check{\sigma}_s\|_1 &= (\Sigma_{11}\Sigma_{22})^{\frac{s-1}{2}} \left| \bar{\sigma}'_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) \right| + \\ &\quad \frac{(\Sigma_{11} + \Sigma_{22})(\Sigma_{11}\Sigma_{22})^{\frac{s}{2}-1}}{2} \left| s \bar{\sigma}_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) - \bar{\sigma}'_s \left( \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right) \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}} \right|. \end{aligned}$$

For simplicity, let  $\tilde{\rho} = \frac{\Sigma_{12}}{\sqrt{\Sigma_{11}\Sigma_{22}}}$ . For  $s = 1$ , Arora *et al.* [13] showed that the above expression is  $1 + o(\gamma)$ . We consider the case for  $s > 1$ . Using Eqn. (C.11), we have,

$$\begin{aligned}\|\nabla \check{\sigma}_s\|_1 &= \frac{s^2}{2s-1} (\Sigma_{11}\Sigma_{22})^{\frac{s-1}{2}} |\bar{\sigma}_{s-1}(\tilde{\rho})| + \frac{(\Sigma_{11} + \Sigma_{22})(\Sigma_{11}\Sigma_{22})^{\frac{s}{2}-1}}{2} \left| s\bar{\sigma}_s(\tilde{\rho}) - \frac{s^2}{2s-1} \bar{\sigma}_{s-1}(\tilde{\rho})\tilde{\rho} \right| \\ &\leq \frac{2s}{3} (\Sigma_{11}\Sigma_{22})^{\frac{s-1}{2}} + \frac{3s}{4} (\Sigma_{11} + \Sigma_{22})(\Sigma_{11}\Sigma_{22})^{\frac{s}{2}-1} \\ &\leq \frac{13s}{6} (1 + \gamma)^{s-1}\end{aligned}$$

If  $\gamma \leq 1/s$ , then  $\|\nabla \check{\sigma}_s\|_1 \leq 6s$ , as required.

### C.2.3 Proof of Lemma C.1.12

We have,

$$\begin{aligned}&\left| c_s \frac{\text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}'))}{d_l} \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle - \prod_{j=l}^L \dot{\Sigma}^{(j)}(\mathbf{y}, \mathbf{y}') \right| \\ &\leq \left| c_s \frac{\text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}'))}{d_l} - \dot{\Sigma}^{(l)}(\mathbf{y}, \mathbf{y}') \right| \left| \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle \right| + \\ &\quad \left| \dot{\Sigma}^{(l)}(\mathbf{y}, \mathbf{y}') \right| \left| \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle - \prod_{j=l+1}^L \dot{\Sigma}^{(j)}(\mathbf{y}, \mathbf{y}') \right| \\ &\leq \left| \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle \right| \varepsilon_4 + \frac{s^2}{2s-1} \check{\sigma}_{s-1}(\Lambda^{(l)}(\mathbf{x}, \mathbf{x}')) \varepsilon_2\end{aligned}$$

Since  $\bar{\mathcal{B}}^{l+1}(\varepsilon_2)$ ,  $\langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle \leq \prod_{j=l+1}^L \dot{\Sigma}^{(j)}(\mathbf{y}, \mathbf{y}') + \varepsilon_2 \leq \prod_{j=l+1}^L \dot{\Sigma}^{(j)}(\mathbf{y}, \mathbf{y}') + 1$ .

Moreover, since  $\dot{\Sigma}^{(l)}(\mathbf{y}, \mathbf{y}') = \frac{s^2}{2s-1} \check{\sigma}_{s-1}(\Lambda^{(l)}(\mathbf{y}, \mathbf{y}')) \leq \frac{s^2}{2s-1}$ , we have,

$$\begin{aligned}&\left| c_s \frac{\text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}'))}{d_l} \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle - \prod_{j=l+1}^L \dot{\Sigma}^{(j)}(\mathbf{y}, \mathbf{y}') \right| \leq \left( \left( \frac{s^2}{2s-1} \right)^{L-l} + 1 \right) \varepsilon_4 + s\varepsilon_2 \\ &\leq s^{L-l} \varepsilon_4 + s\varepsilon_2.\end{aligned}$$

### C.2.4 Proof of Lemma C.1.13

Fix any  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$ . Recall that  $\mathbf{G}^{(l)}(\mathbf{y}, \mathbf{y}') = [\mathbf{h}^{(l)}(\mathbf{y}) \quad \mathbf{h}^{(l)}(\mathbf{y}')] \in \mathbb{R}^{d_l \times 2}$ . Similarly define,  $\mathbf{F}^{(l+1)}(\mathbf{y}, \mathbf{y}') = [\mathbf{f}^{(l+1)}(\mathbf{y}) \quad \mathbf{f}^{(l+1)}(\mathbf{y}')] \in \mathbb{R}^{d_l \times 2}$ . Using Lemma C.1.5 for each row of  $\mathbf{W}^{(l+1)}$ , we can conclude that

$$\mathbf{W}^{(l+1)} \Pi_{\mathbf{G}}^{\perp} \stackrel{d}{=}_{\mathbf{F}^{(l+1)} = \mathbf{W}^{(l+1)} \mathbf{G}^{(l)}} \widetilde{\mathbf{W}} \Pi_{\mathbf{G}}^{\perp},$$

where  $\widetilde{\mathbf{W}}$  is a i.i.d. copy of  $\mathbf{W}^{(l+1)}$ .

Note that  $\left[ \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^{\top} \widetilde{\mathbf{W}} \left( \mathbf{b}^{(l+1)}(\mathbf{y}') \right)^{\top} \widetilde{\mathbf{W}} \right]^{\top} \in \mathbb{R}^{2d_l} \sim \mathcal{N}(0, \Sigma)$  where  $\Sigma = \begin{bmatrix} b \mathbf{I}_{d_l} & b' \mathbf{I}_{d_l} \\ b' \mathbf{I}_{d_l} & b'' \mathbf{I}_{d_l} \end{bmatrix}$ . In the definition of  $\Sigma$ ,  $b = \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}) \rangle$ ,  $b' = \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle$  and  $b'' = \langle \mathbf{b}^{(l+1)}(\mathbf{y}'), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle$ . If  $\mathbf{M}$  is such that  $\Sigma = \mathbf{M} \mathbf{M}^{\top}$ , then

$$\left[ \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^{\top} \widetilde{\mathbf{W}} \left( \mathbf{b}^{(l+1)}(\mathbf{y}') \right)^{\top} \widetilde{\mathbf{W}} \right]^{\top} \stackrel{d}{=} \mathbf{M} \boldsymbol{\xi},$$

where  $\boldsymbol{\xi} \sim \mathcal{N}(0, \mathbf{I}_{2d_l})$ . Thus, for fixed  $\mathbf{G}^{(l)}$  and conditioned on the value of  $\{\mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}'), \mathbf{h}^{(l)}(\mathbf{y}), \mathbf{h}^{(l)}(\mathbf{y}')\}$ , we have,

$$\begin{aligned} & \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^{\top} \mathbf{W}^{(l+1)} \Pi_{\mathbf{G}}^{\perp} \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_{\mathbf{G}}^{\perp} \left( \mathbf{W}^{(l+1)} \right)^{\top} \mathbf{b}^{(l+1)}(\mathbf{y}) \\ & \stackrel{d}{=} \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^{\top} \widetilde{\mathbf{W}} \Pi_{\mathbf{G}}^{\perp} \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_{\mathbf{G}}^{\perp} \left( \widetilde{\mathbf{W}} \right)^{\top} \mathbf{b}^{(l+1)}(\mathbf{y}) \\ & \stackrel{d}{=} ([\mathbf{I}_{d_l} \ 0] \mathbf{M} \boldsymbol{\xi})^{\top} \Pi_{\mathbf{G}}^{\perp} \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_{\mathbf{G}}^{\perp} ([0 \ \mathbf{I}_{d_l}] \mathbf{M} \boldsymbol{\xi}) \\ & \stackrel{d}{=} \frac{1}{2} \boldsymbol{\xi}^{\top} \mathbf{M}^{\top} \begin{bmatrix} 0 & \Pi_{\mathbf{G}}^{\perp} \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_{\mathbf{G}}^{\perp} \\ \Pi_{\mathbf{G}}^{\perp} \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_{\mathbf{G}}^{\perp} & 0 \end{bmatrix} \mathbf{M} \boldsymbol{\xi} \end{aligned}$$

Define

$$\mathbf{A} := \frac{1}{2} \mathbf{M}^{\top} \begin{bmatrix} 0 & \Pi_{\mathbf{G}}^{\perp} \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_{\mathbf{G}}^{\perp} \\ \Pi_{\mathbf{G}}^{\perp} \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_{\mathbf{G}}^{\perp} & 0 \end{bmatrix} \mathbf{M}.$$

Then we have,

$$\begin{aligned}
\mathbb{E}[\xi^\top \mathbf{A} \xi] &= \text{trace}(\mathbf{A}) = \frac{1}{2} \text{trace} \left( \begin{bmatrix} 0 & \mathbf{\Pi}_G^\perp \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \mathbf{\Pi}_G^\perp \\ \mathbf{\Pi}_G^\perp \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \mathbf{\Pi}_G^\perp & 0 \end{bmatrix} \Sigma \right) \\
&= b' \text{trace}(\mathbf{\Pi}_G^\perp \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \mathbf{\Pi}_G^\perp \mathbf{I}_{d_l}) \\
&= b' \text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}') \mathbf{\Pi}_G^\perp) \\
&= b' \text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}') (\mathbf{I}_{d_l} - \mathbf{\Pi}_G)) \\
&= b' [\text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}')) - \text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}') \mathbf{\Pi}_G)].
\end{aligned}$$

Since  $\text{trace}(\mathbf{\Pi}_G) = \text{rank}(\mathbf{\Pi}_G) \leq 2$ , we have,

$$0 \leq \text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}') \mathbf{\Pi}_G) \leq \|\Delta^{(l)}(\mathbf{y}, \mathbf{y}')\|_2 \text{trace}(\mathbf{\Pi}_G) \leq 2 \|\Delta^{(l)}(\mathbf{y}, \mathbf{y}')\|_2.$$

Consequently,

$$|\mathbb{E}[\xi^\top \mathbf{A} \xi] - b' \text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}'))| \leq 2b' \|\Delta^{(l)}(\mathbf{y}, \mathbf{y}')\|_2.$$

For the upper bound on the spectrum, note that  $\|\mathbf{M}\|_2^2 = \|\Sigma\|_2 \leq b + b''$ . Hence,

$$\begin{aligned}
\|\mathbf{A}\|_2 &\leq \frac{1}{2} \|\mathbf{M}\|_2^2 \|\mathbf{\Pi}_G^\perp \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \mathbf{\Pi}_G^\perp\|_2 \\
&\leq \frac{1}{2} (b + b'') \|\mathbf{\Pi}_G^\perp\|_2 \|\Delta^{(l)}(\mathbf{y}, \mathbf{y}')\|_2 \|\mathbf{\Pi}_G^\perp\|_2 \\
&\leq \frac{b + b''}{2} \|\Delta^{(l)}(\mathbf{y}, \mathbf{y}')\|_2.
\end{aligned}$$

Thus, on using Lemma C.2.5 with  $t = \log(3/\delta)$ , we have with probability at least

$$1 - \delta/3$$

$$\begin{aligned}
|\xi^\top \mathbf{A} \xi - \mathbb{E}[\xi^\top \mathbf{A} \xi]| &\leq 2(\|\mathbf{A}\|_F \sqrt{t} + \|\mathbf{A}\|_2 t) \\
&\leq 2\|\mathbf{A}\|_2 (\sqrt{2d_l t} + t) \\
&\leq (b + b'') \|\Delta^{(l)}(\mathbf{y}, \mathbf{y}')\|_2 \left( \sqrt{2d_l \log\left(\frac{3}{\delta}\right)} + \log\left(\frac{3}{\delta}\right) \right).
\end{aligned}$$

Consequently,

$$\begin{aligned}
& \left| \frac{c_s}{d_l} \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \mathbf{\Pi}_G^\perp \boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y}') \mathbf{\Pi}_G^\perp \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}) - \frac{c_s}{d_l} \text{trace}(\boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y}')) \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \rangle \right| \\
& \leq \frac{c_s}{d_l} |\xi^\top \mathbf{A} \xi - \mathbb{E}[\xi^\top \mathbf{A} \xi]| + \frac{c_s}{d_l} |\mathbb{E}[\xi^\top \mathbf{A} \xi] - b' \text{trace}(\boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y}'))| \\
& \leq c_s(b + b'') \|\boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y}')\|_2 \left( \sqrt{\frac{2}{d_l} \log\left(\frac{3}{\delta}\right)} + \frac{1}{d_l} \log\left(\frac{3}{\delta}\right) \right) + \frac{2b'c_s}{d_l} \|\boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y}')\|_2,
\end{aligned}$$

holds with probability at least  $1 - \delta/3$ . The lemma follows by taking a union bound over all possible choices of  $(\mathbf{y}, \mathbf{y}')$ .

We can use the above result to obtain the following bound for any  $\mathbf{y} \in \{\mathbf{x}, \mathbf{x}'\}$

$$\begin{aligned}
& \sqrt{\frac{c_s}{d_l}} \left\| \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \mathbf{\Pi}_G^\perp \mathbf{D}^{(l)}(\mathbf{y}) \right\| \\
& \leq \left\{ \frac{c_s}{d_l} \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \mathbf{\Pi}_G^\perp \boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y}) \mathbf{\Pi}_G^\perp \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}) \right\}^{1/2} \\
& \leq \left\{ \frac{c_s}{d_l} \text{trace}(\boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y}))b + 2c_s b \|\boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y})\|_2 \left( \sqrt{\frac{2}{d_l} \log\left(\frac{3}{\delta}\right)} + \frac{1}{d_l} \log\left(\frac{3}{\delta}\right) \right) + \frac{2bc_s}{d_l} \|\boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y}')\|_2 \right\}^{1/2} \\
& \leq \sqrt{bc_s \|\boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y})\|_2} \cdot \left\{ 1 + 2 \left( \sqrt{\frac{2}{d_l} \log\left(\frac{3}{\delta}\right)} + \frac{1}{d_l} \left( 1 + \log\left(\frac{3}{\delta}\right) \right) \right) \right\}^{1/2} \\
& \leq \sqrt{2bc_s \|\boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y})\|_2},
\end{aligned}$$

where  $b = \langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}) \rangle$  and the last step uses the relation the bound on  $d_l$ .

### C.2.5 Proof of Lemma C.1.14

Fix any  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$ . For ease of writing let  $\mathbf{V}^{(l+1)}(\mathbf{x}) := (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)}(\mathbf{x})$ . We are interested in bounding the following expression.

$$\begin{aligned}
& (\mathbf{V}^{(l+1)}(\mathbf{y}))^\top \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \mathbf{V}^{(l+1)}(\mathbf{y}') - (\mathbf{V}^{(l+1)}(\mathbf{y}))^\top \Pi_G^\perp \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_G^\perp \mathbf{V}^{(l+1)}(\mathbf{y}') \\
&= (\mathbf{V}^{(l+1)}(\mathbf{y}))^\top (\Pi_G + \Pi_G^\perp) \mathbf{D}^{(l)}(\mathbf{y}) \mathbf{D}^{(l)}(\mathbf{y}') (\Pi_G + \Pi_G^\perp) \mathbf{V}^{(l+1)}(\mathbf{y}') - \\
&\quad (\mathbf{V}^{(l+1)}(\mathbf{y}))^\top \Pi_G^\perp \mathbf{D}^{(l)}(\mathbf{y}) \mathbf{D}^{(l)}(\mathbf{y}') \Pi_G^\perp \mathbf{V}^{(l+1)}(\mathbf{y}') \\
&= (\mathbf{V}^{(l+1)}(\mathbf{y}))^\top \Pi_G \mathbf{D}^{(l)}(\mathbf{y}) \mathbf{D}^{(l)}(\mathbf{y}') \Pi_G \mathbf{V}^{(l+1)}(\mathbf{y}') + (\mathbf{V}^{(l+1)}(\mathbf{y}))^\top \Pi_G \mathbf{D}^{(l)}(\mathbf{y}) \mathbf{D}^{(l)}(\mathbf{y}') \Pi_G^\perp \mathbf{V}^{(l+1)}(\mathbf{y}') + \\
&\quad (\mathbf{V}^{(l+1)}(\mathbf{y}))^\top \Pi_G^\perp \mathbf{D}^{(l)}(\mathbf{y}) \mathbf{D}^{(l)}(\mathbf{y}') \Pi_G \mathbf{V}^{(l+1)}(\mathbf{y}').
\end{aligned}$$

Consequently,

$$\begin{aligned}
& \frac{c_s}{d_l} \left| \left( \mathbf{V}^{(l+1)}(\mathbf{y}) \right)^\top \boldsymbol{\Lambda}^{(l)}(\mathbf{y}, \mathbf{y}') \mathbf{V}^{(l+1)}(\mathbf{y}') - \left( \mathbf{V}^{(l+1)}(\mathbf{y}) \right)^\top \boldsymbol{\Pi}_G^\perp \boldsymbol{\Lambda}^{(l)}(\mathbf{y}, \mathbf{y}') \boldsymbol{\Pi}_G^\perp \mathbf{V}^{(l+1)}(\mathbf{y}') \right| \\
&= \frac{c_s}{d_l} \left| \left( \mathbf{V}^{(l+1)}(\mathbf{y}) \right)^\top \boldsymbol{\Pi}_G \mathbf{D}^{(l)}(\mathbf{y}) \mathbf{D}^{(l)}(\mathbf{y}') \boldsymbol{\Pi}_G \mathbf{V}^{(l+1)}(\mathbf{y}') + \left( \mathbf{V}^{(l+1)}(\mathbf{y}) \right)^\top \boldsymbol{\Pi}_G \mathbf{D}^{(l)}(\mathbf{y}) \mathbf{D}^{(l)}(\mathbf{y}') \boldsymbol{\Pi}_G^\perp \mathbf{V}^{(l+1)}(\mathbf{y}') + \right. \\
&\quad \left. \left( \mathbf{V}^{(l+1)}(\mathbf{y}) \right)^\top \boldsymbol{\Pi}_G^\perp \mathbf{D}^{(l)}(\mathbf{y}) \mathbf{D}^{(l)}(\mathbf{y}') \boldsymbol{\Pi}_G \mathbf{V}^{(l+1)}(\mathbf{y}') \right| \\
&\leq \frac{c_s}{d_l} \left\| \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \boldsymbol{\Pi}_G \mathbf{D}^{(l)}(\mathbf{y}) \right\| \left\| \mathbf{D}^{(l)}(\mathbf{y}') \boldsymbol{\Pi}_G \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}') \right\| + \\
&\quad \frac{c_s}{d_l} \left\| \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \boldsymbol{\Pi}_G \mathbf{D}^{(l)}(\mathbf{y}) \right\| \left\| \mathbf{D}^{(l)}(\mathbf{y}') \boldsymbol{\Pi}_G^\perp \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}') \right\| + \\
&\quad \frac{c_s}{d_l} \left\| \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \boldsymbol{\Pi}_G^\perp \mathbf{D}^{(l)}(\mathbf{y}) \right\| \left\| \mathbf{D}^{(l)}(\mathbf{y}') \boldsymbol{\Pi}_G \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}') \right\| \\
&\leq \frac{c_s}{d_l} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{8}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{8}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] \|\mathbf{D}^{(l)}(\mathbf{y})\|_2 \|\mathbf{D}^{(l)}(\mathbf{y}')\|_2 + \\
&\quad c_s \sqrt{\frac{2}{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{8}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] \|\mathbf{D}^{(l)}(\mathbf{y})\|_2 \|\mathbf{b}^{(l+1)}(\mathbf{y}')\| \|\mathbf{D}^{(l)}(\mathbf{y}')\|_2 + \\
&\quad c_s \sqrt{\frac{2}{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{8}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] \|\mathbf{D}^{(l)}(\mathbf{y})\|_2 \|\mathbf{b}^{(l+1)}(\mathbf{y})\| \|\mathbf{D}^{(l)}(\mathbf{y}')\|_2 \\
&\leq \frac{c_s}{d_l} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{8}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right]^2 \|\boldsymbol{\Lambda}^{(l)}(\mathbf{y}, \mathbf{y}')\|_2 + \\
&\quad c_s \sqrt{\frac{8}{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{8}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] (s^{(L-l)/2} + 1) \|\boldsymbol{\Delta}^{(l)}(\mathbf{y}, \mathbf{y}')\|_2 \\
&\leq \frac{c_s \varepsilon_5}{\sqrt{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{8}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] \times \\
&\quad \cdot \left\{ \frac{1}{\sqrt{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{8}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] + \sqrt{8} (s^{(L-l)/2} + 1) \right\},
\end{aligned}$$

where we used Lemmas C.1.13 and C.2.6 in fourth step and the condition of  $\bar{\mathcal{B}}^{l+1}(\varepsilon_2) \wedge \bar{\mathcal{E}}^l(\varepsilon_5)$  in the last step. Both lemmas hold with probability at least  $1 - \delta/2$ , ensuring that the overall expression is true with probability at least  $1 - \delta$ .

### C.2.6 Proof of Lemma C.1.15

Fix any  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$ . Conditioned on  $\bar{\mathcal{A}}^L(\varepsilon_1/2) \wedge \bar{\mathcal{B}}^{l+1}(\varepsilon_2) \wedge \bar{\mathcal{C}}(\varepsilon_3) \wedge \bar{\mathcal{D}}^l(\varepsilon_4) \wedge \bar{\mathcal{E}}^l(\varepsilon_5)$ , we begin with bounding  $\left| \langle \mathbf{b}^{(l)}(\mathbf{y}), \mathbf{b}^{(l)}(\mathbf{y}') \rangle - \prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{x}, \mathbf{y}') \right|$ . Using the definition of  $\mathbf{b}^{(l)}(\mathbf{y})$  we have,

$$\begin{aligned}
& \left| \left\langle \mathbf{b}^{(l)}(\mathbf{y}), \mathbf{b}^{(l)}(\mathbf{y}') \right\rangle - \prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{y}, \mathbf{y}') \right| \\
&= \left| \frac{c_s}{d_l} \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}') - \prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{y}, \mathbf{y}') \right| \\
&\leq \frac{c_s}{d_l} \left| \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}') \right. \\
&\quad \left. - \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \Pi_G^\perp \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_G^\perp \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}') \right| \\
&\quad + \left| \frac{c_s}{d_l} \left( \mathbf{b}^{(l+1)}(\mathbf{y}) \right)^\top \mathbf{W}^{(l+1)} \Pi_G^\perp \Delta^{(l)}(\mathbf{y}, \mathbf{y}') \Pi_G^\perp \left( \mathbf{W}^{(l+1)} \right)^\top \mathbf{b}^{(l+1)}(\mathbf{y}) - \right. \\
&\quad \left. \frac{c_s}{d_l} \text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}')) \left\langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \right\rangle \right| \\
&\quad + \left| c_s \frac{\text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}'))}{d_l} \left\langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \right\rangle - \prod_{j=l}^{L+1} \dot{\Sigma}^{(j)}(\mathbf{y}, \mathbf{y}') \right| \\
&\leq \left( \frac{c_s \varepsilon_5}{\sqrt{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{16}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] \times \right. \\
&\quad \left. \left\{ \frac{1}{\sqrt{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{16}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] + \sqrt{8} (s^{(L-l)/2} + 1) \right\} \right) \\
&\quad + c_s \left( \left\langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}) \right\rangle + \left\langle \mathbf{b}^{(l+1)}(\mathbf{y}'), \mathbf{b}^{(l+1)}(\mathbf{y}') \right\rangle \right) \left( \sqrt{\frac{2}{d_l} \log \left( \frac{6}{\delta} \right)} + \frac{1}{d_l} \log \left( \frac{6}{\delta} \right) \right) \varepsilon_5 \\
&\quad + \frac{2c_s}{d_l} \left\langle \mathbf{b}^{(l+1)}(\mathbf{y}), \mathbf{b}^{(l+1)}(\mathbf{y}') \right\rangle \varepsilon_5 + s^{L-l} \varepsilon_4 + s \varepsilon_2 \\
&\leq \left( \frac{c_s \varepsilon_5}{\sqrt{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{16}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] \times \right. \\
&\quad \left. \left\{ \frac{1}{\sqrt{d_l}} \left[ (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{16}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2} \right] + \sqrt{8} (s^{(L-l)/2} + 1) \right\} \right) \\
&\quad + 2c_s (s^{L-l} + 1) \left( \sqrt{\frac{2}{d_l} \log \left( \frac{6}{\delta} \right)} + \frac{1}{d_l} \log \left( \frac{6}{\delta} \right) \right) \varepsilon_5 + \frac{2c_s}{d_l} (s^{L-l} + 1) \varepsilon_5 + s^{L-l} \varepsilon_4 + s \varepsilon_2.
\end{aligned}$$

In the fourth step, we used Lemmas C.1.12, C.1.14 and C.1.13 with the last two each holding with probability at least  $1 - \delta/2$ . Consequently, the above expression holds with probability at least  $1 - \delta$ .

### C.2.7 Proof of Lemma C.2.2

We carry out the analysis conditioned on  $\bar{\mathcal{A}}^l(\varepsilon)$ . Since  $\left| \left( \mathbf{h}^{(l)}(\mathbf{y}) \right)^\top \mathbf{h}^{(l)}(\mathbf{y}') - \Sigma^{(l)}(\mathbf{y}, \mathbf{y}') \right| \leq \varepsilon$  for all  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$  we can conclude that  $\|\tilde{\mathbf{G}}^{(l)} - \Lambda^{(l)}\|_\infty \leq 2\varepsilon$ . Since  $\check{\sigma}_{s-1}$  is  $\beta_{s-1}$ -Lipschitz in  $\mathcal{M}_+^{\gamma_{s-1}}$  w.r.t.  $\infty$ -norm, we have,

$$|\check{\sigma}_{s-1}(\mathbf{G}^{(l)}) - \check{\sigma}_{s-1}(\Lambda^{(l+1)})| \leq \beta_{s-1} \|\tilde{\mathbf{G}}^{(l)} - \Lambda^{(l+1)}\|_\infty \leq 2\beta_{s-1}\varepsilon,$$

for  $\varepsilon \leq \gamma_{s-1}$ . Fix any  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$ . We have that,

$$\begin{aligned} \left| c_s \frac{\text{trace}(\Delta^{(l+1)}(\mathbf{y}, \mathbf{y}'))}{d_{l+1}} - \dot{\Sigma}^{(l+1)}(\mathbf{y}, \mathbf{y}') \right| &\leq \left| c_s \frac{\text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}'))}{d_l} - \frac{s^2}{2s-1} \check{\sigma}_{s-1}(\Lambda^{(l+1)}(\mathbf{y}, \mathbf{y}')) \right| \\ &\leq \left| c_s \frac{\text{trace}(\Delta^{(l)}(\mathbf{y}, \mathbf{y}'))}{d_l} - \frac{s^2}{2s-1} \check{\sigma}_{s-1}(\tilde{\mathbf{G}}^{(l)}(\mathbf{y}, \mathbf{y}')) \right| + \\ &\quad \frac{s^2}{2s-1} |\check{\sigma}_{s-1}(\tilde{\mathbf{G}}^{(l)}) - \check{\sigma}_{s-1}(\Lambda^{(l+1)})| \\ &\leq s^2(1+\varepsilon)^{s-1} \varphi_{s-1}(d_l, \delta/2, \min\{1, \Sigma^{(l)}(\mathbf{y}, \mathbf{y}') + \varepsilon\}) + \frac{2s^2\beta_s\varepsilon}{2s-1}, \end{aligned}$$

holds with probability  $1 - \delta$ . The last step follows from Lemma C.2.4. On taking a union bound, we can conclude that

$$\left| c_s \frac{\text{trace}(\Delta^{(l+1)}(\mathbf{y}, \mathbf{y}'))}{d_{l+1}} - \dot{\Sigma}^{(l+1)}(\mathbf{y}, \mathbf{y}') \right| \leq s^2(1+\varepsilon)^{s-1} \varphi_{s-1}(d_l, \delta/6, 1) + \frac{2s^2\beta_s\varepsilon}{2s-1},$$

holds for all  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$  with probability  $1 - \delta$ . Since we have already invoked Lemma C.2.4, it also gives us that

$$\|\Delta^{(l+1)}(\mathbf{x}, \mathbf{x}')\|_2 \leq s^2 \left[ 2(1+\varepsilon) \log \left( \frac{6d_l}{\delta} \right) \right]^{s-1},$$

holds for all  $(\mathbf{y}, \mathbf{y}') \in \{(\mathbf{x}, \mathbf{x}), (\mathbf{x}, \mathbf{x}'), (\mathbf{x}', \mathbf{x}')\}$  with probability  $1 - \delta$ . Using this result along with Lemma C.1.4 and  $\varepsilon \leq 1/s$  yields us

$$\Pr \left[ \bar{\mathcal{A}}^l(\varepsilon) \Rightarrow \bar{\mathcal{D}}^{l+1} \left( 3s^2 \varphi_{s-1}(d_l, \delta/6, 1) + \frac{2s^2\beta_s\varepsilon}{2s-1} \right) \wedge \bar{\mathcal{E}}^{l+1} \left( 3s^2 \left[ 2 \log \left( \frac{6d_l}{\delta} \right) \right]^{s-1} \right) \right] \geq 1 - \delta,$$

as required.

### C.2.8 Proof of Lemma C.2.3

Consider,

$$\begin{aligned}
\Pr[\bar{\mathcal{A}}(\varepsilon) \Rightarrow \bar{\mathcal{D}}(\varepsilon') \wedge \bar{\mathcal{E}}(\varepsilon'')] &= \Pr\left[\left(\bigcap_{l=0}^L \bar{\mathcal{A}}^l(\varepsilon)\right) \Rightarrow \left(\bigcap_{l=0}^L \{\bar{\mathcal{D}}^{l+1}(\varepsilon') \wedge \bar{\mathcal{E}}^{l+1}(\varepsilon'')\}\right)\right] \\
&= \Pr\left[\neg\left(\bigcap_{l=0}^L \bar{\mathcal{A}}^l(\varepsilon)\right) \vee \left(\bigcap_{l=0}^L \{\bar{\mathcal{D}}^{l+1}(\varepsilon') \wedge \bar{\mathcal{E}}^{l+1}(\varepsilon'')\}\right)\right] \\
&= 1 - \Pr\left[\left(\bigcap_{l=0}^L \bar{\mathcal{A}}^l(\varepsilon)\right) \wedge \left(\bigcup_{l=0}^L \neg\{\bar{\mathcal{D}}^{l+1}(\varepsilon') \wedge \bar{\mathcal{E}}^{l+1}(\varepsilon'')\}\right)\right] \\
&= 1 - \Pr\left[\left(\bigcup_{l=0}^L \left(\bigcap_{l=0}^L \bar{\mathcal{A}}^l(\varepsilon)\right) \wedge \neg\{\bar{\mathcal{D}}^{l+1}(\varepsilon') \wedge \bar{\mathcal{E}}^{l+1}(\varepsilon'')\}\right)\right] \\
&\geq 1 - \sum_{l=0}^L \Pr\left[\left(\bigcap_{l=0}^L \bar{\mathcal{A}}^l(\varepsilon)\right) \wedge \neg\{\bar{\mathcal{D}}^{l+1}(\varepsilon') \wedge \bar{\mathcal{E}}^{l+1}(\varepsilon'')\}\right] \\
&\geq 1 - \sum_{l=0}^L \Pr\left[\bar{\mathcal{A}}^l(\varepsilon) \wedge \neg\{\bar{\mathcal{D}}^{l+1}(\varepsilon') \wedge \bar{\mathcal{E}}^{l+1}(\varepsilon'')\}\right] \\
&\geq 1 - \sum_{l=0}^L \Pr\left[\neg(\bar{\mathcal{A}}^l(\varepsilon) \Rightarrow \bar{\mathcal{D}}^{l+1}(\varepsilon') \wedge \bar{\mathcal{E}}^{l+1}(\varepsilon''))\right] \\
&\geq 1 - \sum_{l=0}^L \frac{\delta}{L+1} \\
&\geq 1 - \delta,
\end{aligned}$$

where the eighth line follows from Lemma C.2.2 and the choice of  $\varepsilon'$  and  $\varepsilon''$ .

### C.2.9 Proof of Lemma C.2.4

We can rewrite  $\text{trace}(\Delta^{(l)}(\mathbf{x}, \mathbf{x}'))$  as follows:

$$\begin{aligned}\text{trace}(\Delta^{(l)}(\mathbf{x}, \mathbf{x}')) &= \text{trace}[\mathbf{D}^{(l)}(\mathbf{x})\mathbf{D}^{(l)}(\mathbf{x}')] \\ &= \text{trace}[\text{diag}(\sigma'_s(\mathbf{f}^{(l)}(\mathbf{x})))\text{diag}(\sigma'_s(\mathbf{f}^{(l)}(\mathbf{x}')))] \\ &= \langle \sigma'_s(\mathbf{f}^{(l)}(\mathbf{x})), \sigma'_s(\mathbf{f}^{(l)}(\mathbf{x}')) \rangle \\ &= s^2 \langle \sigma_{s-1}(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)}(\mathbf{x})), \sigma_{s-1}(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)}(\mathbf{x}')) \rangle.\end{aligned}$$

Note that since all the matrices  $\{\mathbf{W}^{(i)}\}_{i=1}^{l-1}$  are fixed,  $\mathbf{h}^{(l-1)}(\cdot)$  is a fixed function. Consequently, each term of the vector  $\mathbf{W}^{(l)}\mathbf{h}^{(l-1)}(\mathbf{x})$  is distributed as  $\mathcal{N}(0, \langle \mathbf{h}^{(l)}(\mathbf{x}), \mathbf{h}^{(l)}(\mathbf{x}) \rangle)$  and is independent of others. If  $(\mathbf{w}_j^{(l)})^\top$  denotes the  $j^{\text{th}}$  row of the matrix  $\mathbf{W}^{(l)}$ , then we can write the inner product as

$$\begin{aligned}\text{trace}(\Delta^{(l)}(\mathbf{x}, \mathbf{x}')) &= s^2 \langle \sigma_{s-1}(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)}(\mathbf{x})), \sigma_{s-1}(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)}(\mathbf{x}')) \rangle \\ &= s^2 \sum_{j=1}^{d_l} \sigma_{s-1}\left((\mathbf{w}_j^{(l)})^\top \mathbf{h}^{(l-1)}(\mathbf{x})\right) \sigma_{s-1}\left((\mathbf{w}_j^{(l)})^\top \mathbf{h}^{(l-1)}(\mathbf{x}')\right) \\ &= s^2 \sum_{j=1}^{d_l} Z_j,\end{aligned}$$

where  $Z_j$  are all independent and identically distributed as  $\sigma_{s-1}(X)\sigma_{s-1}(Y)$  where  $(X, Y) \sim \mathcal{N}(0, \tilde{\mathbf{G}}^{(l-1)}(\mathbf{x}, \mathbf{x}'))$ . If we define  $\tilde{Z}_j := (G_{11}G_{22})^{-\frac{(s-1)}{2}}$ , then  $\tilde{Z}_j$  are all independent and identically distributed as  $\sigma_{s-1}(\tilde{X})\sigma_{s-1}(\tilde{Y})$  where  $(\tilde{X}, \tilde{Y}) \sim \mathcal{N}(0, \Sigma_{\rho_G})$  and  $\rho_G = \frac{G_{12}}{\sqrt{G_{11}G_{22}}}$ . From Lemma C.1.7, we know that

$$\left| \frac{1}{d_l} \sum_{j=1}^{d_l} \tilde{Z}_j - \mathbb{E}[\tilde{Z}_1] \right| \leq \varphi_{s-1}(d_l, \delta, \rho_G)$$

with probability at least  $1 - \delta$ . Consequently,

$$\left| \frac{1}{d_l} \sum_{j=1}^{d_l} Z_j - \mathbb{E}[Z_1] \right| \leq (G_{11}G_{22})^{\frac{(s-1)}{2}} \varphi_{s-1}(d_l, \delta, \rho_G)$$

with probability at least  $1 - \delta$ . Note that  $\mathbb{E}[Z_1] = \mathbb{E}_{(X,Y) \sim \mathcal{N}(0, \tilde{\mathbf{G}}^{(l)}(\mathbf{x}, \mathbf{x}'))} [\sigma_{s-1}(X)\sigma_{s-1}(Y)] = \frac{1}{c_{s-1}} \check{\sigma}_{s-1}(\tilde{\mathbf{G}}^{(l)}(\mathbf{x}, \mathbf{x}'))$  and  $c_{s-1} = c_s(2s - 1)$ . On combining this with the previous result, we can conclude that,

$$\begin{aligned} \left| c_s \frac{\text{trace}(\Delta^{(l)}(\mathbf{x}, \mathbf{x}'))}{d_l} - \frac{s^2}{2s-1} \check{\sigma}_{s-1}(\mathbf{G}^{(l-1)}(\mathbf{x}, \mathbf{x}')) \right| &\leq s^2 \left| \frac{1}{d_l} \sum_{j=1}^{d_l} Z_j - \mathbb{E}[Z_1] \right| \\ &\leq s^2 (G_{11} G_{22})^{\frac{(s-1)}{2}} \varphi_{s-1}(d_l, \delta/2, \rho_G), \end{aligned}$$

holds with probability  $1 - \delta/2$ . For the spectral norm of  $\Delta^{(l)}(\mathbf{x}, \mathbf{x}')$ , we have,

$$\|\Delta^{(l)}(\mathbf{x}, \mathbf{x}')\|_2 = s^2 \max_{j \in d_l} Z_j = s^2 (G_{11} G_{22})^{\frac{(s-1)}{2}} \max_{j \in d_l} \tilde{Z}_j.$$

Using Eqn. (C.17), we can show that

$$\begin{aligned} \Pr\left(\max_{j \in d_l} \tilde{Z}_j > \left[(1 + \rho_G) \log\left(\frac{d_l}{\delta}\right)\right]^{s-1}\right) &\leq d_l \Pr\left(\tilde{Z}_1 > \left[(1 + \rho_G) \log\left(\frac{d_l}{\delta}\right)\right]^{s-1}\right) \\ &\leq d_l \exp\left(-\frac{1 + \rho_G}{1 + \rho_G} \log\left(\frac{d_l}{\delta}\right)\right) \leq \delta. \end{aligned}$$

Consequently,

$$\begin{aligned} \|\Delta^{(l)}(\mathbf{x}, \mathbf{x}')\|_2 &= s^2 (G_{11} G_{22})^{\frac{(s-1)}{2}} \max_{j \in d_l} \tilde{Z}_j \\ &\leq s^2 (G_{11} G_{22})^{\frac{(s-1)}{2}} \left[\left(1 + \rho_G \log\left(\frac{2d_l}{\delta}\right)\right)\right]^{s-1} \end{aligned}$$

holds with probability  $1 - \delta/2$ . Thus, both the hold simultaneously with probability at least  $1 - \delta$ .

### C.2.10 Proof of Lemma C.2.6

We will prove the claim for  $\mathbf{y} \in \{\mathbf{x}, \mathbf{x}'\}$ . Recall that  $\mathbf{G}^{(l)}(\mathbf{y}, \mathbf{y}') = [\mathbf{h}^{(l)}(\mathbf{y}) \ \mathbf{h}^{(l)}(\mathbf{y}')] \in \mathbb{R}^{d_l \times 2}$ . For simplicity, we drop the arguments from  $\mathbf{h}^{(l)}$  and  $\mathbf{b}^{(l+1)}$ . Define  $\mathbf{\Pi_h} = \mathbf{h}\mathbf{h}^\top$  and  $\mathbf{\Pi_{G/h}} = \mathbf{\Pi_G} - \mathbf{\Pi_h}$ . Note that  $\mathbf{\Pi_{G/h}}$  is still a projection matrix of rank 0 or 1.

Recall that  $\mathbf{b}^{(l+1)}$  is the gradient with respect to the pre-activation layer  $l + 1$  given by  $\mathbf{f}^{(l+1)} = \mathbf{W}^{(l+1)}\mathbf{h}^{(l)}$ . If we view the output  $f$  as a function of  $\mathbf{h}^{(l)}, \mathbf{W}^{(l+1)}, \dots, \mathbf{W}^{(L+1)}$ , then we have the following relation:

$$\frac{\partial}{\partial \mathbf{h}^{(l)}} f(\mathbf{h}^{(l)}, \mathbf{W}^{(l+1)}, \dots, \mathbf{W}^{(L+1)}) = (\mathbf{b}^{(l+1)})^\top \mathbf{W}^{(l+1)}.$$

Recall that  $\sigma_s$  is  $s$ -homogeneous, that is,  $\sigma_s(\lambda x) = \lambda^s \sigma_s(x)$  for any  $\lambda > 0$ . Using this recursion repeatedly, we have,

$$\begin{aligned} f(\lambda \mathbf{h}^{(l)}, \mathbf{W}^{(l+1)}, \dots, \mathbf{W}^{(L+1)}) &= \lambda^{s^{L-l}} f(\mathbf{h}^{(l)}, \mathbf{W}^{(l+1)}, \dots, \mathbf{W}^{(L+1)}) \\ \implies \frac{\partial}{\partial \lambda} f(\lambda \mathbf{h}^{(l)}, \mathbf{W}^{(l+1)}, \dots, \mathbf{W}^{(L+1)}) &= s^{L-l} \lambda^{(s^{L-l}-1)} f(\mathbf{h}^{(l)}, \mathbf{W}^{(l+1)}, \dots, \mathbf{W}^{(L+1)}). \end{aligned}$$

Using this, we can write,

$$\begin{aligned} f(\mathbf{h}^{(l)}, \mathbf{W}^{(l+1)}, \dots, \mathbf{W}^{(L+1)}) &= s^{L-L} \frac{\partial}{\partial \lambda} f(\lambda \mathbf{h}^{(l)}, \mathbf{W}^{(l+1)}, \dots, \mathbf{W}^{(L+1)}) \Big|_{\lambda=1} \\ &= s^{L-L} \frac{\partial}{\partial (\lambda \mathbf{h}^{(l)})} f(\lambda \mathbf{h}^{(l)}, \mathbf{W}^{(l+1)}, \dots, \mathbf{W}^{(L+1)}) \Big|_{\lambda=1} \frac{\partial (\lambda \mathbf{h}^{(l)})}{\partial \lambda} \Big|_{\lambda=1} \\ &= s^{L-L} \frac{\partial}{\partial (\lambda \mathbf{h}^{(l)})} f(\lambda \mathbf{h}^{(l)}, \mathbf{W}^{(l+1)}, \dots, \mathbf{W}^{(L+1)}) \Big|_{\lambda=1} \mathbf{h}^{(l)} \\ &= s^{L-L} (\mathbf{b}^{(l+1)})^\top \mathbf{W}^{(l+1)} \mathbf{h}^{(l)}. \end{aligned}$$

By definition of  $\Pi_h$ , we have,

$$\begin{aligned} \left\| \Pi_h (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)} \right\| &= \left\| \frac{1}{\|\mathbf{h}^{(l)}\|^2} \cdot \mathbf{h}^{(l)} (\mathbf{h}^{(l)})^\top (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)} \right\| \\ &= s^{L-l} \frac{|f(\mathbf{y}; \mathbf{W})|}{\|\mathbf{h}^{(l)}\|}. \end{aligned}$$

Since  $\bar{\mathcal{A}}^L(\varepsilon_1/2) \wedge \bar{C}(\varepsilon_3)$ ,  $\|\mathbf{h}^{(l)}\| \geq 1 - \varepsilon_1/2 \geq 1/2$  and  $|f(\mathbf{y}; \mathbf{W})| \leq \varepsilon_3$ . Consequently,

$$\left\| \Pi_h (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)} \right\| \leq s^{L-l} \varepsilon_3 \sqrt{2}.$$

Similar to the proof of Lemma C.1.13, we note that for a fixed  $\mathbf{h}$  and conditioned on  $\mathbf{f}^{(l+1)}$  and  $\mathbf{b}^{(l+1)}$  Lemma C.1.5 gives us that

$$\mathbf{W}^{(l+1)} \Pi_h^\perp \stackrel{d}{=}_{\mathbf{f}^{(l+1)} = \mathbf{W}^{(l+1)} \mathbf{h}^{(l)}} \widetilde{\mathbf{W}} \Pi_h^\perp,$$

where  $\widetilde{\mathbf{W}}$  is a i.i.d. copy of  $\mathbf{W}^{(l+1)}$ . From the definition of  $\boldsymbol{\Pi}_{\mathbf{G}/\mathbf{h}}$ , we can conclude that  $\boldsymbol{\Pi}_{\mathbf{G}/\mathbf{h}}^\top \boldsymbol{\Pi}_{\mathbf{h}} = 0$ . Thus, combining this with the previous equation, we can conclude that

$$\mathbf{W}^{(l+1)} \boldsymbol{\Pi}_{\mathbf{G}/\mathbf{h}} \xrightarrow{d_{\mathbf{f}^{(l+1)} = \mathbf{W}^{(l+1)} \mathbf{h}^{(l)}}} \widetilde{\mathbf{W}} \boldsymbol{\Pi}_{\mathbf{G}/\mathbf{h}}.$$

If  $\text{rank}(\boldsymbol{\Pi}_{\mathbf{G}/\mathbf{h}}) = 1$ , then  $\boldsymbol{\Pi}_{\mathbf{G}/\mathbf{h}} = \mathbf{u}\mathbf{u}^\top$  for some unit vector  $\mathbf{u}$ . Thus,

$$\begin{aligned} \left\| \boldsymbol{\Pi}_{\mathbf{G}/\mathbf{h}} (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)} \right\| &= \left\| \mathbf{u}\mathbf{u}^\top (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)} \right\| \\ &= \left| \mathbf{u}^\top (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)} \right| \\ &\stackrel{d}{=} \left| \mathbf{u}^\top \widetilde{\mathbf{W}}^\top \mathbf{b}^{(l+1)} \right| = |t|, \end{aligned}$$

where  $t \sim \mathcal{N}(0, \|\mathbf{b}^{(l+1)}\|^2)$ . Therefore, with probability at least  $1 - \delta/2$ ,

$$|t| \leq \|\mathbf{b}^{(l+1)}\| \sqrt{2 \log \left( \frac{4}{\delta} \right)} \leq (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{4}{\delta} \right)}.$$

In the above step, we used similar arguments as used in Appendix C.2.7 to bound  $\|\mathbf{b}^{(l+1)}\|$ . If  $\text{rank}(\boldsymbol{\Pi}_{\mathbf{G}/\mathbf{h}}) = 0$ ,  $\left\| \boldsymbol{\Pi}_{\mathbf{G}/\mathbf{h}} (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)} \right\| = 0$ . On combining this with the previous result, we obtain that with probability  $1 - \delta$

$$\begin{aligned} \left\| \boldsymbol{\Pi}_{\mathbf{G}} (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)} \right\| &\leq \left\| \boldsymbol{\Pi}_{\mathbf{G}/\mathbf{h}} (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)} \right\| + \left\| \boldsymbol{\Pi}_{\mathbf{h}} (\mathbf{W}^{(l+1)})^\top \mathbf{b}^{(l+1)} \right\| \\ &\leq (s^{(L-l)/2} + 1) \sqrt{2 \log \left( \frac{4}{\delta} \right)} + s^{L-l} \varepsilon_3 \sqrt{2}. \end{aligned}$$

Taking the union bound over the two possible choices of  $\mathbf{y}$  gives us the final result.

### C.3 Proof of Theorem 4.3.3

The proof of Theorem 4.3.1 involves combining several smaller results, some of which follow from Theorem 4.3.1 and the others are minor modifications of

existing results. We begin with stating these results along with proofs, if needed, and then combine them to prove Theorem 4.3.1.

Firstly, using Theorem 4.3.1 along with Lemma 5.1 from [346], we can conclude that for  $m \geq \text{poly}(T, L, K, \lambda^{-1}, \lambda_0^{-1}, S^{-1}, \log(1/\delta))$ , there exists a  $\mathbf{W}^*$  such that  $h(\mathbf{x}) = \langle \mathbf{g}(\mathbf{x}; \mathbf{W}_0), \mathbf{W}^* - \mathbf{W}_0 \rangle$  with probability at least  $1 - \delta$  for all  $\mathbf{x} \in \mathcal{Z} = \{\{\mathbf{x}_{t,a}\}_{a=1}^K\}_{t=1}^T$ . Furthermore,  $\|\mathbf{W}^* - \mathbf{W}_0\|_2 \leq S$ .

Secondly, using Lemma C.1.6 and C.1.10, along with Eqn. (C.3), we can conclude that for  $m \geq \text{poly}(T, L, K, \lambda^{-1}, \lambda_0^{-1}, S^{-1}, \log(1/\delta))$ ,  $\|\mathbf{g}(\mathbf{x}; \mathbf{W}_0)\| \leq O(s^L)$ , independent of  $m$ .

Thirdly, we know that the spectral norm of the Hessian of the neural net is  $O(m^{-1/2})$  [196, Theorem 3.2] for all  $\mathbf{W}$  such that  $\|\mathbf{W} - \mathbf{W}_0\| \leq R$ , where  $R$  is some finite radius. Here the implied constant depends on  $s, L$  and  $R$ . Thus, by choosing  $m$  to be sufficiently large, we can conclude that for all  $\|\mathbf{W} - \mathbf{W}_0\| \leq \sqrt{T/\lambda}$ ,  $\|\tilde{\mathbf{H}}(\mathbf{W})\| \leq C_{s,L}m^{-1/3}$ . Here  $\tilde{\mathbf{H}}$  denotes the Hessian and  $C_{s,L}$  denotes a constant that depends only on  $s$  and  $L$ . Using this relation, we can conclude that for any  $\|\mathbf{W} - \mathbf{W}_0\| \leq \sqrt{T/\lambda}$ , we have

$$\begin{aligned} |f(\mathbf{x}; \mathbf{W}) - f(\mathbf{x}; \mathbf{W}_0) - \langle \mathbf{g}(\mathbf{x}; \mathbf{W}_0), \mathbf{W} - \mathbf{W}_0 \rangle| &\leq C_{s,L}m^{-1/3}T/\lambda \\ \implies |f(\mathbf{x}; \mathbf{W}) - \langle \mathbf{g}(\mathbf{x}; \mathbf{W}_0), \mathbf{W} - \mathbf{W}_0 \rangle| &\leq C_{s,L}\lambda^{-1}m^{-1/6}, \end{aligned}$$

where the last step follows by choosing a sufficiently large  $m$  and Assumption 4.2.1. Moreover, the above result holds for any  $\mathbf{x} \in \mathcal{Z}$ .

Lastly, using the relation  $h(\mathbf{x}) = \langle \mathbf{g}(\mathbf{x}; \mathbf{W}_0), \mathbf{W}^* - \mathbf{W}_0 \rangle$  along with the result from Vakili *et al.* [299, Theorem 1] on the kernel defined by neural net at initial-

ization, i.e.  $\hat{k}(\mathbf{x}; \mathbf{x}') = \langle \mathbf{g}(\mathbf{x}; \mathbf{W}_0), \mathbf{g}(\mathbf{x}'; \mathbf{W}_0) \rangle$ , we can conclude that

$$|h(\mathbf{x}) - \mathbf{g}^\top(\mathbf{x}; \mathbf{W}_0)\mathbf{V}_t^{-1}\mathbf{r}| \leq \left( S + \nu \sqrt{\frac{2}{\lambda} \log(1/\delta)} \right) \hat{\sigma}_t(\mathbf{x}).$$

In the above equation,  $\mathbf{V}_t$  and  $\hat{\sigma}_t$  are defined with respect to the dataset  $\mathcal{D}$  and  $\mathbf{r} = \sum_{i=1}^t y_i \mathbf{g}(\mathbf{x}_i; \mathbf{W}_0)$ . Also, we have used the independence of dataset from noise sequence to invoke the above theorem.

Using these results we can prove our main result. For any  $\mathbf{x} \in \mathcal{Z}$ , we have,

$$\begin{aligned} |h(\mathbf{x}) - f(\mathbf{x}; \mathbf{W}_t)| &\leq |h(\mathbf{x}) - \mathbf{g}^\top(\mathbf{x}; \mathbf{W}_0)\mathbf{V}^{-1}\mathbf{r}| + |\mathbf{g}^\top(\mathbf{x}; \mathbf{W}_0)\mathbf{V}^{-1}\mathbf{r} - f(\mathbf{x}; \mathbf{W}_t)| \\ &\leq \left( S + \nu \sqrt{\frac{2}{\lambda} \log(1/\delta)} \right) \hat{\sigma}_t(\mathbf{x}) + |\mathbf{g}^\top(\mathbf{x}; \mathbf{W}_0)\mathbf{V}_t^{-1}\mathbf{r} - f(\mathbf{x}; \mathbf{W}_t)| \\ &\leq \frac{C_{s,L}}{\lambda m^{1/6}} + \left( S + \nu \sqrt{\frac{2}{\lambda} \log(1/\delta)} \right) \hat{\sigma}(\mathbf{x}) + |\mathbf{g}^\top(\mathbf{x}; \mathbf{W}_0)\mathbf{V}_t^{-1}\mathbf{r} - \langle \mathbf{g}(\mathbf{x}; \mathbf{W}_0), \mathbf{W}_t - \mathbf{W}_0 \rangle| \\ &\leq \frac{C_{s,L}}{\lambda m^{1/6}} + \left( S + \nu \sqrt{\frac{2}{\lambda} \log(1/\delta)} \right) \hat{\sigma}(\mathbf{x}) + \|\mathbf{W}_t - \mathbf{W}_0 - \mathbf{V}_t^{-1}\mathbf{r}\| \|\mathbf{V}_t\| \hat{\sigma}(\mathbf{x}) \\ &\leq \frac{C_{s,L}}{\lambda m^{1/6}} + \left( S + \nu \sqrt{\frac{2}{\lambda} \log(1/\delta)} + (1 - \eta\lambda)^{J/2} \sqrt{t/\lambda} + \frac{C'_{s,L}}{\lambda m^{1/6}} \right) (\lambda + C''_{s,L} t) \hat{\sigma}(\mathbf{x}), \end{aligned}$$

as required. As before  $C_{s,L}$ ,  $C'_{s,L}$  and  $C''_{s,L}$  represent constants that depend only on  $s$  and  $L$ . In the last but one step, we used the fact that for any positive definite matrix  $\mathbf{A} \in \mathbb{R}^{d \times d}$  and  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ ,  $\langle \mathbf{x}, \mathbf{y} \rangle \leq (\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x}) \cdot (\mathbf{y}^\top \mathbf{A} \mathbf{y}) \leq (\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x}) \|\mathbf{A}\|_2 \|\mathbf{y}\|_2$ . And in the last step, we used the results from [346, Lemmas B.3, B.5, C.2] along with the bound on gradient established earlier.

## C.4 Additional Details on NeuralGCB

We first provide all the pseudo codes for NeuralGCB followed by proof of Theorem 4.4.1.

### C.4.1 Pseudo Codes

---

**Algorithm 25** NeuralGCB

---

```

1: Require: Time horizon  $T$ , maximum initial variance  $\sigma_0$ , error probability  $\delta$ 
2: Initialize:  $R \leftarrow \lceil \log_2 T \rceil$ , Ensemble of  $R$  Neural Nets with  $\mathbf{W}_0^{(r)} = \mathbf{W}_0$ ,  $\Psi_0^{(r)} \leftarrow \emptyset$ ,  $\forall r \in [R]$ ,  $\mathcal{H} \leftarrow \emptyset$ , arrays  $\text{ctr}$ ,  $\text{max\_mu}$ ,  $\text{fb}$  of size  $R$  with all elements set to 0, batch sizes  $\{q_r\}_{r=1}^R$ 
3: for  $t = 1, 2, 3, \dots, T$  do
4:    $r \leftarrow 1, \hat{A}_r(t) = [K]$ 
5:   while True do
6:     Receive the context-action pairs  $\{\mathbf{x}_{t,a}\}_{a=1}^K$ 
7:      $\{f(\mathbf{x}_{t,a}; \mathbf{W}_t^{(r)}), \hat{\sigma}_{t-1}^{(r)}(\mathbf{x}_{t,a})\}_{a=1}^K, \mathbf{W}_t^{(r)}, \text{fb}[r] \leftarrow \text{GetPredictions}(\mathcal{H}, \Psi_{t-1}^{(r)}, \{\mathbf{x}_{t,a}\}_{a=1}^K, \text{fb}[r], \mathbf{W}_{t-1}^{(r)}, q_r)$ 
8:      $\tilde{\sigma}_{t-1}^{(r)} \leftarrow \max_{a \in \hat{A}_r(t)} \hat{\sigma}_{t-1}^{(r)}(\mathbf{x}_{t,a}), \text{max\_mu}[r] \leftarrow \arg \max_{a \in \hat{A}_r(t)} f(\mathbf{x}_{t,a}; \mathbf{W}_{t-1}^{(r)})$ 
9:     if  $\tilde{\sigma}_{t-1}^{(r)} \leq \sigma_0 2^{-r}$  then
10:       $a_{\text{UCB}} \leftarrow \arg \max_{a \in \hat{A}_r(t)} \text{UCB}_{t-1}^{(r)}(\mathbf{x}_{t,a})$ 
11:      if  $\sigma_{t-1}^{(r)}(x_{t,a_{\text{UCB}}}) \leq \eta_0 / \sqrt{t}$  then
12:        Choose  $a_t \leftarrow a_{\text{UCB}}$  and set  $v_t \leftarrow 1$ 
13:        Receive  $y_t = h(\mathbf{x}_{t,a_t}) + \xi_t$  and update  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(x_{t,a_t}, y_t)\}$ 
14:        Set  $\Psi_t^{(r+1)} \leftarrow \Psi_{t-1}^{(r+1)} \cup \{(t, v_t)\}$  and  $\Psi_t^{(r')} \leftarrow \Psi_{t-1}^{(r')}$  for all  $r' \in [R] \setminus \{r+1\}$ 
15:        break
16:      else
17:         $\hat{A}_{r+1}(t) \leftarrow \{a \in \hat{A}_r(t) : \text{UCB}_{t-1}^{(r)}(\mathbf{x}_{t,a}) \geq \max_{a' \in \hat{A}_r(t)} \text{LCB}_{t-1}^{(r)}(\mathbf{x}_{t,a'})\}, r \leftarrow r + 1$ 
18:      end if
19:    else
20:      if  $r = 1$  or  $\text{ctr}[r] > \alpha_0 4^r$  then
21:        Choose any  $a_t \in \hat{A}_r(t)$  such that  $\hat{\sigma}_{t-1}^{(r)}(x_{t,a}) > \sigma_0 2^{-r}$  and set  $v_t \leftarrow 2$ 
22:      else
23:        Choose  $a_t \leftarrow \text{max\_mu}[r - 1]$  and set  $v_t \leftarrow 3$ 
24:      end if
25:      Receive  $y_t = h(\mathbf{x}_{t,a_t}) + \xi_t$  and update  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(x_{t,a_t}, y_t)\}, \text{ctr}[r] \leftarrow \text{ctr}[r] + 1$ 
26:      Set  $\Psi_t^{(r)} \leftarrow \Psi_{t-1}^{(r)} \cup \{(t, v_t)\}$  and  $\Psi_t^{(r')} \leftarrow \Psi_{t-1}^{(r')}$  for all  $r' \in [R] \setminus \{r\}$ 
27:      break
28:    end if
29:  end while
30: end for

```

---

In Alg. 25,  $\text{UCB}_t^{(r)}$  and  $\text{LCB}_t^{(r)}$  refer to the upper and lower confidence scores respectively at time  $t$  corresponding to index  $r$  and are defined as  $\text{UCB}_t^{(r)}(\cdot) = f(\cdot; \mathbf{W}_t^{(r)}) + \beta_t \hat{\sigma}_t^{(r)}(\cdot)$  and  $\text{LCB}_t^{(r)}(\cdot) = f(\cdot; \mathbf{W}_t^{(r)}) - \beta_t \hat{\sigma}_t^{(r)}(\cdot)$ . The arrays  $\text{ctr}$ ,  $\text{max\_mu}$  and

$t_{fb}$  are used to store the exploitation count, maximizer of the neural net output and feedback time instants respectively. The exploitation count is tracked to ensure that the exploitation budget is not exceeded. Similarly, the maximizer of the neural net output is stored to guide the algorithm in exploitation at the next level and the feedback time instants are used to keep track of last time instant when the neural net was retrained, or equivalently obtained a feedback. Lastly,  $v_t$ 's provide analytical and notational convenience and are not crucial to the working of the algorithm.

GetPredictions is a local routine that calculates the predictive mean and variance after appropriately retraining the neural net and described in Alg. 26 and used as a sub-routine in NeuralGCB. We would like to emphasize that NeuralGCB computes only the mean with the trained parameters. The predictive variance is computed using the gradients at initialization. This is made possible by having an additional ad hoc neural net which is just used to compute the gradients for calculating the variance. This is similar to the setup used in Kassraie and Krause [162].

After computing the predictive variance for the set of current actions, the GetPredictions routine decides whether the neural net needs to be retrained based on the previous feedback instant  $t_{fb}$  and batch parameter  $q$ . In particular, we consider two training schemes, fixed and adaptive. The fixed frequency scheme retrains the neural net if there are an additional of  $q$  points in  $\Psi$  after  $t_{fb}$ . Consequently, the neural corresponding to the  $r^{\text{th}}$  subset in the partition is retrained after every  $q_r$  points are added to the subset. On the other hand, in the adaptive scheme, inspired by the rarely switching bandits [1, 318], the neural net is retrained whenever  $\det(\mathbf{V}) > q \det(\mathbf{V}_{fb})$  where  $\mathbf{V}$  and  $\mathbf{V}_{fb}$  are as defined in

line 2 and 3 of Alg. 26. Since  $\log(\det(\mathbf{V}) / \det(\mathcal{U}))$  is a measure of information gain, the neural net is effectively retrained only once sufficient additional information has been obtained since the last time the neural net was trained. We would like to point out that in the main text, we only refer to the fixed training scheme due to space constraints. However, our NeuralGCB works even with the adaptive scheme and the regret guarantees under this training schedule are formalized in later in the section.

Lastly, TrainNN is another local routine that carries out the training of neural net for a given dataset using gradient descent for  $J$  epochs with a step size of  $\eta$  starting with  $\mathbf{W}_0$ .

---

**Algorithm 26** GetPredictions

---

- 1: **Input:** Set of past actions and their corresponding rewards  $\mathcal{H}$ , Index set  $\Psi$ , Current set of actions  $\{\mathbf{x}_{t,a}\}_{a=1}^K$ , Previous feedback instant  $t_{\text{fb}}$ ,  $\mathbf{W}_{t_{\text{fb}}}$ , Batch choice parameter  $q$
  - 2:  $\mathbf{Z} \leftarrow \lambda I + \frac{1}{m} \sum_{i \in \Psi} \mathbf{g}(\mathbf{x}_i; \mathbf{W}_0) \mathbf{g}(\mathbf{x}_i; \mathbf{W}_0)^\top$
  - 3:  $\mathbf{Z}_{\text{fb}} \leftarrow \lambda I + \frac{1}{m} \sum_{i \in \Psi, i \leq t_{\text{fb}}} \mathbf{g}(\mathbf{x}_i; \mathbf{W}_0) \mathbf{g}(\mathbf{x}_i; \mathbf{W}_0)^\top$
  - 4: Set  $\sigma^2(\mathbf{x}_{t,a}) \leftarrow \mathbf{g}(\mathbf{x}_{t,a}; \mathbf{W}_0)^\top \mathbf{Z}^{-1} \mathbf{g}(\mathbf{x}_{t,a}; \mathbf{W}_0) / m$  for all  $a \in [K]$
  - 5: **For fixed batch size:**  $\text{to\_train} = (|\Psi| - t_{\text{fb}} == q)$
  - 6: **For adaptive batch size:**  $\text{to\_train} = (\det(\mathbf{Z}) > q \det(\mathbf{Z}_{\text{fb}}))$
  - 7: **if**  $\text{to\_train}$  **then**
  - 8:    $\mathbf{W} \leftarrow \text{TrainNN}(m, L, J, \eta, \lambda, \mathbf{W}_0, \{(x_r, y_r) \in \mathcal{H} : r \in \Psi\})$ ,  $t_{\text{fb}} \leftarrow |\Psi|$
  - 9: **else**
  - 10:    $\mathbf{W} \leftarrow \mathbf{W}_{t_{\text{fb}}}$
  - 11: **end if**
  - 12: **return**  $\{f(\mathbf{x}_{t,a}; \mathbf{W}), \sigma(\mathbf{x}_{t,a})\}_{a=1}^K, \mathbf{W}, t_{\text{fb}}$ .
- 

---

**Algorithm 27** TrainNN( $m, L, J, \eta, \lambda, \mathbf{W}_0, \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ )

---

- 1: Define  $\mathcal{L}(\mathbf{W}) = \sum_{i=1}^n (f(\mathbf{x}_i; \mathbf{W}) - y_i)^2 + m\lambda\|\mathbf{W} - \mathbf{W}_0\|^2$
  - 2: **for**  $j = 0, 1, \dots, J - 1$  **do**
  - 3:    $\mathbf{W}_{j+1} = \mathbf{W}_j - \eta \nabla_{\mathbf{W}} \mathcal{L}(\mathbf{W}_j)$
  - 4: **end for**
  - 5: **return**  $\mathbf{W}_J$
-

### C.4.2 Proof of Theorem 4.4.1

To analyse the regret of NeuralGCB, we divide the time horizon into three disjoint sets depending on how the action at that time instant was chosen. In particular, for  $i = \{1, 2, 3\}$ , we define

$$\begin{aligned}\mathcal{T}_i(s) &:= \{t \in \psi_T^{(r)} : v_t = i\} \\ \mathcal{T}_i &:= \bigcup_{r=1}^R \mathcal{T}_i(r).\end{aligned}$$

Thus,  $\mathcal{T}_1, \mathcal{T}_2$  and  $\mathcal{T}_3$  consist of all the points chosen by the chosen algorithm at line 12, 21 and 23 respectively. We consider the regret incurred at time instants in each of these sets separately. We begin with  $\mathcal{T}_1$ . For any  $t \in \mathcal{T}_1(r)$ ,

$$\begin{aligned}h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t}) &\leq f(\mathbf{x}_{t,a_t^*}; \mathbf{W}_{t-1}^{(r)}) + \tilde{\beta} + \beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t^*}) - (f(\mathbf{x}_{t,a_t}; \mathbf{W}_{t-1}^{(r)}) - \tilde{\beta} - \beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t})) \\ &\leq f(\mathbf{x}_{t,a_t}; \mathbf{W}_{t-1}^{(r)}) + \tilde{\beta} + \beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t}) - f(\mathbf{x}_{t,a_t}; \mathbf{W}_{t-1}^{(r)}) + \tilde{\beta} + \beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t}) \\ &\leq 2\tilde{\beta} + 2\beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t}) \\ &\leq 2\tilde{\beta} + \frac{2\eta_0 \beta_{t-1}}{\sqrt{t}},\end{aligned}$$

where we use Theorem 4.3.3 in the first step,  $\tilde{\beta} = \frac{C_{s,L}}{\lambda m^{1/6}}$  and  $\beta_t = \left(S + \nu \sqrt{\frac{2}{\lambda} \log(1/\delta)} + (1 - \eta\lambda)^{J/2} \sqrt{t/\lambda} + \frac{C'_{s,L}}{\lambda m^{1/6}}\right)(\lambda + C''_{s,L} t)$ . Since this is independent of  $r$ , this relation holds for all  $t \in \mathcal{T}_1$ . Consequently, the regret incurred over the time instants in  $\mathcal{T}_1$  can be bounded as

$$\begin{aligned}\sum_{t \in \mathcal{T}_1} h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t}) &\leq \sum_{t \in \mathcal{T}_1} \left[ 2\tilde{\beta} + \frac{2\eta_0 \beta_{t-1}}{\sqrt{t}} \right] \\ &\leq 2\tilde{\beta} |\mathcal{T}_1| + 2\eta_0 \beta_T \sqrt{T}.\end{aligned}$$

We now consider a time instant in  $\mathcal{T}_2$  or  $\mathcal{T}_3$ . Fix any  $r > 1$  and  $t \in \mathcal{T}_2(r) \cup \mathcal{T}_3(r)$  which implies that  $a_t \in \hat{A}_r(t)$ . Consequently,  $\mathbf{x}_{t,a_t}$  satisfies the following relation:

$$f(\mathbf{x}_{t,a_t}; \mathbf{W}_{t-1}^{(r-1)}) + \beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t}) \geq \max_{a' \in \hat{A}_r(t)} f(\mathbf{x}_{t,a'}; \mathbf{W}_{t-1}^{(r-1)}) - \beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a'})$$

Note that output of the neural net is based on the parameters evaluated during the last time the neural net was trained, which we denote by  $t_{\text{fb}}$ . In other words,  $f(\mathbf{x}_{t,a_t}; \mathbf{W}_{t-1}^{(r-1)}) = f(\mathbf{x}_{t,a_t}; \mathbf{W}_{t_{\text{fb}}}^{(r-1)})$ . However,  $\hat{\sigma}_{t-1}^{(r-1)}$  is evaluated using all the points in  $\Psi_{t-1}^{(r)}$ . We can account for the discrepancy using Lemma 12 from [1] which states that for any two positive definite matrices  $\mathbf{A} \succeq \mathbf{B} \in \mathbb{R}^{d \times d}$  and  $\mathbf{x} \in \mathbb{R}^d$ , we have  $\mathbf{x}^\top \mathbf{A} \mathbf{x} \leq \mathbf{x}^\top \mathbf{B} \mathbf{x} \cdot \sqrt{\det(\mathbf{A}) / \det(\mathbf{B})}$ . Using this result along with noting that  $\hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a}) = \mathbf{g}^\top(\mathbf{x}_{t,a}; \mathbf{W}_0) \mathbf{V}_{t-1}^{-1} \mathbf{g}(\mathbf{x}_{t,a}; \mathbf{W}_0)$ ,  $\hat{\sigma}_{t_{\text{fb}}}^{(r-1)}(\mathbf{x}_{t,a}) = \mathbf{g}^\top(\mathbf{x}_{t,a}; \mathbf{W}_0) \mathbf{V}_{t_{\text{fb}}}^{-1} \mathbf{g}(\mathbf{x}_{t,a}; \mathbf{W}_0)$  and  $\mathbf{V}_{t-1} \succeq \mathbf{V}_{\text{fb}}$ , we can conclude that  $\hat{\sigma}_{t_{\text{fb}}}^{(r-1)}(\mathbf{x}_{t,a}) \leq \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a}) \cdot \sqrt{\det(\mathbf{V}_{t-1}) / \det(\mathbf{V}_{\text{fb}})}$ .

For any batch such that  $\det(\mathbf{V}_t) / \det(\mathbf{V}_{\text{fb}}) \leq 4$ , we have,  $\hat{\sigma}_{t_{\text{fb}}}^{(r-1)}(\mathbf{x}_{t,a}) \leq 2\hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a})$  for all  $a \in [K]$ . Consequently,  $|h(\mathbf{x}_{t,a}) - f(\mathbf{x}_{t,a_t}; \mathbf{W}_{t_{\text{fb}}}^{(r-1)})| \leq \tilde{\beta} + \beta_{t_{\text{fb}}} \hat{\sigma}_{t_{\text{fb}}}^{(r-1)}(\mathbf{x}_{t,a}) \leq \tilde{\beta} + 2\beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a})$ . Thus, for any such batch and a time instant  $t \in \mathcal{T}_2(r)$ , using Theorem 4.3.3 we have,

$$\begin{aligned} f(\mathbf{x}_{t,a_t}; \mathbf{W}_{t-1}^{(r-1)}) + \beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t}) &\geq \max_{a' \in \hat{A}_r(t)} f(\mathbf{x}_{t,a'}; \mathbf{W}_{t-1}^{(r-1)}) - \beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a'}) \\ \implies f(\mathbf{x}_{t,a_t}; \mathbf{W}_{t_{\text{fb}}}^{(r-1)}) + \beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t}) &\geq f(\mathbf{x}_{t,a_t^*}; \mathbf{W}_{t_{\text{fb}}}^{(r-1)}) - \beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a^*}) \\ \implies h(\mathbf{x}_{t,a_t}) + 3\beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t}) + \tilde{\beta} &\geq h(\mathbf{x}_{t,a^*}) - 3\beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a^*}) - \tilde{\beta}. \end{aligned}$$

Since  $\hat{A}_r(t) \subseteq \hat{A}_{r-1}(t)$ ,  $\hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a}) \leq \tilde{\sigma}_{t-1}^{(r-1)} \leq \sigma_0 2^{1-r}$  for all  $a \in \hat{A}_r(t)$ . Thus,

$$h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t}) \leq 3\beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a}) + 3\beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t^*}) + 2\tilde{\beta} \leq 12\sigma_0 \beta_{t-1} \cdot 2^{-r} + 2\tilde{\beta}.$$

Similarly, for a time instant  $t \in \mathcal{T}_3(r)$ , we have,

$$\begin{aligned} h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t}) &\leq f(\mathbf{x}_{t,a_t^*}; \mathbf{W}_{t_{\text{fb}}}^{(r-1)}) + \tilde{\beta} + 2\beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t^*}) - f(\mathbf{x}_{t,a_t}; \mathbf{W}_{t_{\text{fb}}}^{(r-1)}) + \tilde{\beta} + 2\beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t}) \\ &\leq 3\beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t^*}) + 3\beta_{t-1} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a}) + 2\tilde{\beta} \leq 12\sigma_0 \beta_{t-1} \cdot 2^{-r} + 2\tilde{\beta} \\ &\leq 12\sigma_0 \beta_{t-1} \cdot 2^{-r} + 2\tilde{\beta}, \end{aligned}$$

where the first step uses Theorem 4.3.3 and the last step uses the fact that  $\mathbf{x}_{t,a_t} \in \hat{A}_{r-1}(t)$ . For any  $t \in \mathcal{T}_2(1) \cup \mathcal{T}_3(1)$ , we use the trivial bound  $h(\mathbf{x}_{t,a^*}) - h(\mathbf{x}_{t,a_t}) \leq 2$ .

We now bound  $|\mathcal{T}_2(r)|$  using similar techniques outlined in [308, 14]. Fix any  $r \geq 1$ . Using the fact that for all  $t \in \mathcal{T}_2(r)$ ,  $\hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t}) \geq \sigma_0 2^{-r}$  and the bound on the sum of posterior standard deviations from [68, Lemma 4], we can write

$$\begin{aligned}\sigma_0 2^{-r} |\mathcal{T}_2(r)| &\leq \sum_{t \in \mathcal{T}_2(r)} \hat{\sigma}_{t-1}^{(r-1)}(\mathbf{x}_{t,a_t}) \leq \sqrt{8|\mathcal{T}_2(r)|(\lambda\Gamma_k(T) + 1)} \\ \implies |\mathcal{T}_2(r)| &\leq 2^r \sqrt{8\sigma_0^{-2}|\mathcal{T}_2(r)|(\lambda\Gamma_k(T) + 1)}\end{aligned}$$

We also used the fact that the information gain of the kernel induced by the finite neural net is close to that of NTK for sufficiently large  $m$  [162, Lemma D.5]. This also provides a guideline for setting the length of the exploration sequence. Specifically, we can set  $\alpha_0$  such that  $|\mathcal{T}_3(r)|$  also satisfies  $|\mathcal{T}_3(r)| \leq 2^r \sqrt{8\sigma_0^{-2}|\mathcal{T}_3(r)|(\lambda\Gamma_k(T) + 1)}$ , or equivalently,  $\alpha_0 = O(\Gamma_k(T))$ . In general, we have  $|\mathcal{T}_3(r)| \leq \alpha_0 4^r \implies |\mathcal{T}_3(r)| \leq 2^r \sqrt{\alpha_0 |\mathcal{T}_3(r)|}$ .

We can use these conditions to bound the regret incurred for  $t \in \mathcal{T}'_2$ , the subset of  $\mathcal{T}_2$  that consists only of batches that satisfy the determinant condition. We have,

$$\begin{aligned}\sum_{t \in \mathcal{T}'_2} h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t}) &\leq \sum_{r=1}^R \sum_{t \in \mathcal{T}'_2(r)} h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t}) \\ &\leq |\mathcal{T}_2(1)| + \sum_{r=2}^R \sum_{t \in \mathcal{T}_2(r)} [12\sigma_0 \beta_{t-1} \cdot 2^{-r} + 2\tilde{\beta}] \\ &\leq |\mathcal{T}_2(1)| + \sum_{r=2}^R [12\sigma_0 \beta_{t-1} \cdot 2^{-r} + 2\tilde{\beta}] |\mathcal{T}_2(r)| \\ &\leq |\mathcal{T}_2(1)| + 2\tilde{\beta} |\mathcal{T}_2| + \sum_{r=2}^R (12\sigma_0 \beta_{t-1} \cdot 2^{-r}) \cdot 2^r \sqrt{8\sigma_0^{-2}|\mathcal{T}_2(r)|(\lambda\Gamma_k(T) + 1)} \\ &\leq |\mathcal{T}_2(1)| + 2\tilde{\beta} |\mathcal{T}_2| + 12\beta_T \sqrt{8|\mathcal{T}_2|R(\lambda\Gamma_k(T) + 1)},\end{aligned}$$

where we used Cauchy-Schwarz inequality in the last step. Using a similar analysis, we can bound the regret incurred for  $t \in \mathcal{T}'_3$ , which is defined similarly

to  $\mathcal{T}'_2$ , to obtain,

$$\sum_{t \in \mathcal{T}'_3} h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t}) \leq |\mathcal{T}_3(1)| + 2\tilde{\beta}|\mathcal{T}_3| + 12\beta_T \sqrt{8|\mathcal{T}_3|R\alpha_0}.$$

Now, for batches for which the determinant condition is not satisfied, the regret incurred in any batch corresponding to subset index  $r$  can be trivially bounded as  $q_r$ . We show that the number of such batches is bounded by  $O(\Gamma_k(T))$ . Fix a  $r \geq 1$ . Let there be  $B_r$  such batches for which the determinant condition is not satisfied and  $\mathcal{T}'(r)$  denote the set of time instants at which such batches begin. Then,

$$4^{B_r} \leq \prod_{t \in \mathcal{T}'(r)} \frac{\det(\mathbf{V}_{t+q_r})}{\det(\mathbf{V}_t)} \leq \frac{\det(\mathbf{V}_{T+1})}{\det(\lambda I)}.$$

Since  $\log(\det(\mathbf{V}_{T+1})/\det(\lambda I))$  is  $O(\Gamma_k(T))$ , we can conclude that  $B_r$  is also  $O(\Gamma_k(T))$ . Thus, regret incurred in all such batches can be bounded as  $O(\max_r q_r \Gamma_k(T) \log T)$  as there are  $R = \log T$  batches.

We can combine the two cases to obtain the regret incurred over  $\mathcal{T}_2$  and  $\mathcal{T}_3$  as

$$\begin{aligned} \sum_{t \in \mathcal{T}_2} h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t}) &\leq |\mathcal{T}_2(1)| + 2\tilde{\beta}|\mathcal{T}_2| + 12\beta_T \sqrt{8|\mathcal{T}_2|R(\lambda\Gamma_k(T) + 1)} + O(\max_r q_r \Gamma_k(T) \log T) \\ \sum_{t \in \mathcal{T}_3} h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t}) &\leq |\mathcal{T}_3(1)| + 2\tilde{\beta}|\mathcal{T}_3| + 12\beta_T \sqrt{8|\mathcal{T}_3|R\alpha_0} + O(\max_r q_r \Gamma_k(T) \log T). \end{aligned}$$

The overall regret is now obtained by adding the regret incurred over  $\mathcal{T}_1, \mathcal{T}_2$

and  $\mathcal{T}_3$ . We have,

$$\begin{aligned}
R(T) &= \sum_{t=1}^T h(\mathbf{x}_{t,a^*}) - h(\mathbf{x}_{t,a_t}) \\
&= \sum_{t \in \mathcal{T}_1} h(\mathbf{x}_{t,a^*}) - h(\mathbf{x}_{t,a_t}) + \sum_{t \in \mathcal{T}_2} h(\mathbf{x}_{t,a^*}) - h(\mathbf{x}_{t,a_t}) + \sum_{t \in \mathcal{T}_3} h(\mathbf{x}_{t,a^*}) - h(\mathbf{x}_{t,a_t}) \\
&\leq 2\tilde{\beta}(|\mathcal{T}_1| + |\mathcal{T}_2| + |\mathcal{T}_3|) + |\mathcal{T}_2(1)| + |\mathcal{T}_3(1)| + 2\eta_0\beta_T \sqrt{T} + 12\beta_T \sqrt{8|\mathcal{T}_3|R\alpha_0} \\
&\quad + 12\beta_T \sqrt{8|\mathcal{T}_2|R(\lambda\Gamma_k(T) + 1)} + O(\max_r q_r \Gamma_k(T) \log T) \\
&\leq 2\tilde{\beta}T + 2\eta_0\beta_T \sqrt{T} + 12\beta_T \sqrt{8T(\lambda\Gamma_k(T) + 1 + \alpha_0)} + O(\max_r q_r \Gamma_k(T) \log T),
\end{aligned}$$

where we again use the Cauchy Schwarz inequality in the last step along with the fact that  $|\mathcal{T}_2(1)|$  and  $|\mathcal{T}_3(1)|$  satisfy  $O(\Gamma_k(T))$ . Since  $m \geq \text{poly}(T)$ , the first term  $\tilde{\beta}T$  is  $O(1)$  and using the values of  $J$  and  $\eta$  specified in Assumption 4.2.2 along with the bound on  $m$ , we have  $\beta_T \leq \beta \leq 2S + \nu \sqrt{2\lambda^{-1} \log(1/\delta)}$ . Consequently, the regret incurred by NeuralGCB satisfies  $O(\sqrt{T\Gamma_k(T)\log(1/\delta)} + \sqrt{T\Gamma_k(T)} + \sqrt{T\log(1/\delta)} + \max_r q_r \Gamma_k(T) \log T)$ .

The above analysis carries through almost as is for the case of adaptive batch sizes. Since the batches always satisfy the determinant condition with  $q_r$  instead of 4, we do not need to separately consider the case where the determinant condition is not met. Using the same steps as above, we can conclude that the regret incurred by NeuralGCB when run with adaptive step sizes is  $O(\sqrt{T\Gamma_k(T)\log(1/\delta)} \cdot \max_r \sqrt{q_r})$ . Thus for the adaptive case, we incur an additional multiplicative factor. This can be explained by the fact that in the adaptive batch setting, the number of times the neural nets are retrained is  $O(\Gamma_k(T))$ . However, in the fixed batch setting, even if we use the optimal batch size of  $\sqrt{T/\Gamma_k(T)}$ , we end up retraining the neural nets about  $O(\sqrt{T\Gamma_k(T)})$  times, which is more than in the case of the adaptive batch setting. The more frequent of retraining of neural nets helps us achieve a tighter regret bound in the case of

fixed batch setting.

APPENDIX D  
CHAPTER 5

## D.1 Analysis for PLS

Before providing the detailed proofs of the Theorems regarding the performance of PLS, we state and prove two supplementary lemmas that will be used in the proofs.

**Lemma D.1.1.** *Let  $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n$  be a collection of  $n$  i.i.d. random vectors in  $\mathbb{R}^d$  with mean  $\boldsymbol{\mu}$ . Each coordinate of every random vector is assumed to be an independent sub-Gaussian random variable with variance proxy  $\sigma^2$ . Let  $\{\mathbf{Y}_i\}_{i=1}^n$  be another collection of random vectors such that  $\mathbf{Y}_i = \mathbf{X}_i \mathbb{1}\{x : \|x\| \leq R + B\}$  for all  $i \in \{1, 2, \dots, n\}$ . Then the following relation holds for any  $t > 0, R > \sigma \sqrt{2 \log(4n)}$  and  $B > \|\boldsymbol{\mu}\|_\infty$ ,*

$$\Pr\left(\left\|\frac{1}{n} \sum_{i=1}^n \mathbf{Y}_i - \boldsymbol{\mu}\right\| \geq t\right) \leq 2 \cdot 5^d \cdot \exp\left(-\frac{nt^2}{8\sigma^2}\right).$$

Consequently,  $\left\|\frac{1}{n} \sum_{i=1}^n \mathbf{Y}_i - \boldsymbol{\mu}\right\| \leq \frac{2\sigma}{\sqrt{n}}(2\sqrt{d} + \sqrt{2 \log(2/\delta)})$  with probability at least  $1 - \delta$ .

*Proof.* Let  $\mathbf{v} \in \mathbb{R}^d$  be any unit vector. Since the entries of  $\mathbf{X}_i$  are independent,  $Z_i = \mathbf{v}^\top \mathbf{X}$  is a sub-Gaussian random variable with mean  $\mu_Z = \mathbf{v}^\top \boldsymbol{\mu}$  and variance proxy  $\sigma^2$  for all  $i \in \{1, 2, \dots, n\}$ . Let  $W_i = Z_i \mathbb{1}\{[-(R + B), (R + B)]\}$ . Since  $B \geq \|\boldsymbol{\mu}\|_\infty$ ,  $\mu_Z \leq B$ . We also define the event  $\mathcal{A} := \bigcap_{i=1}^n \mathcal{A}_i$  where  $\mathcal{A}_i = \{|Z_i| \leq R + B\}$ . For any

$\lambda \in \mathbb{R}$ , we have,

$$\begin{aligned}
\mathbb{E} \left[ \exp \left( \lambda \left( \frac{1}{n} \sum_{i=1}^n W_i - \mu_Z \right) \right) \right] &= \int_{-\infty}^{\infty} \exp \left( \lambda \left( \frac{1}{n} \sum_{i=1}^n w_i - \mu_Z \right) \right) f_{W_1, \dots, W_n}(w_1, \dots, w_n) dw_1 \dots dw_n \\
&= \int_{-\infty}^{\infty} \exp \left( \lambda \left( \frac{1}{n} \sum_{i=1}^n w_i - \mu_Z \right) \right) \frac{1}{\Pr(\mathcal{A})} f_{Z_1, \dots, Z_n}(w_1, \dots, w_n) \mathbb{1}\{\mathcal{A}\} dw_1 \dots dw_n \\
&\leq \int_{-\infty}^{\infty} \exp \left( \lambda \left( \frac{1}{n} \sum_{i=1}^n w_i - \mu_Z \right) \right) \frac{1}{\Pr(\mathcal{A})} f_{Z_1, \dots, Z_n}(w_1, \dots, w_n) dw_1 \dots dw_n \\
&\leq \frac{1}{\Pr(\mathcal{A})} \mathbb{E} \left[ \exp \left( \lambda \left( \frac{1}{n} \sum_{i=1}^n Z_i - \mu_Z \right) \right) \right] \\
&\leq \frac{\exp(\lambda^2 \sigma^2 / 2n)}{\Pr(\mathcal{A})}.
\end{aligned}$$

Let us bound the term  $\Pr(\mathcal{A})$ . We have,

$$\begin{aligned}
\Pr(\mathcal{A}) &= \Pr \left( \bigcap_{i=1}^n \mathcal{A}_i \right) \\
&= \Pr \left( \bigcap_{i=1}^n \{|Z_i| \leq R + B\} \right) \\
&= 1 - \Pr \left( \bigcup_{i=1}^n |Z_i| > R + B \right) \\
&\geq 1 - n \Pr(|Z_1| > R + B),
\end{aligned}$$

Since  $Z_1$  is a sub-Gaussian random variable,

$$\begin{aligned}
\Pr(Z_1 > R + B) &= \Pr(Z_1 - \mu_Z > R + B - \mu_Z) \\
&\leq \Pr(Z_1 - \mu_Z > R) \\
&\leq \exp \left( -\frac{R^2}{2\sigma^2} \right).
\end{aligned}$$

Similarly,

$$\begin{aligned}
\Pr(Z_1 < -(R + B)) &= \Pr(Z_1 - \mu_Z < -R - B - \mu_Z) \\
&\leq \Pr(Z_1 - \mu_Z < -R) \\
&\leq \exp \left( -\frac{R^2}{2\sigma^2} \right).
\end{aligned}$$

On combining the two, we obtain,

$$\Pr(\mathcal{A}) \geq 1 - 2n \exp\left(-\frac{R^2}{2\sigma^2}\right).$$

If  $R \geq \sigma \sqrt{2 \log(4n)}$ , then  $\Pr(\mathcal{A}) \geq 1/2$ . Consequently,

$$\mathbb{E} \left[ \exp \left( \lambda \left( \frac{1}{n} \sum_{i=1}^n W_i - \mu_Z \right) \right) \right] \leq 2 \exp(\lambda^2 \sigma^2 / 2n),$$

and

$$\Pr \left( \frac{1}{n} \sum_{i=1}^n W_i - \mu_Z > t \right) \leq 2 \exp(-nt^2/2\sigma^2).$$

To obtain concentration bounds for  $\left\| \frac{1}{n} \sum_{i=1}^n \mathbf{Y}_i - \boldsymbol{\mu} \right\|$ , we use the same technique used for unbounded sub-Gaussian random variables [151]. We reproduce the proof here for completeness. For brevity of notation, we define  $\mathbf{W} := \frac{1}{n} \sum_{i=1}^n \mathbf{Y}_i$ .

Let  $\mathcal{N}$  denote a minimal  $1/2$ -cover of  $\mathcal{B}_d(1)$ , where  $\mathcal{B}_d(r) = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 \leq r\}$ . This implies that for all  $\mathbf{x} \in \mathcal{B}_d(1)$ ,  $\exists \mathbf{y} \in \mathcal{N}$  such that  $\|\mathbf{x} - \mathbf{y}\|_2 \leq 1/2$ . Using the standard volumetric bounds, the cardinality of  $\mathcal{N}$  can be bounded as  $|\mathcal{N}| \leq 5^d$ .

Once again, let  $\mathbf{v} \in \mathcal{B}_d(1)$  denote a unit vector. For every  $\mathbf{v}$ , we have a  $\mathbf{z}_v \in \mathcal{N}$  such that  $\mathbf{v} = \mathbf{z}_v + \mathbf{u}$  with  $\|\mathbf{u}\| \leq 1/2$ . Thus, for any vector  $\mathbf{X}$ , we have,

$$\begin{aligned} \max_{\mathbf{v} \in \mathcal{B}_d(1)} \mathbf{v}^\top \mathbf{X} &= \max_{\mathbf{v} \in \mathcal{B}_d(1)} (\mathbf{z}_v + \mathbf{u})^\top \mathbf{X} \\ &\leq \max_{\mathbf{z} \in \mathcal{N}} \mathbf{z}^\top \mathbf{X} + \max_{\mathbf{u} \in \mathcal{B}_d(1/2)} \mathbf{u}^\top \mathbf{X} \\ &\leq \max_{\mathbf{z} \in \mathcal{N}} \mathbf{z}^\top \mathbf{X} + \frac{1}{2} \max_{\mathbf{u} \in \mathcal{B}_d(1)} \mathbf{u}^\top \mathbf{X}. \end{aligned}$$

Thus,  $\max_{\mathbf{v} \in \mathcal{B}_d(1)} \mathbf{v}^\top \mathbf{X} \leq 2 \max_{\mathbf{z} \in \mathcal{N}} \mathbf{z}^\top \mathbf{X}$ . Moreover, since  $\|\mathbf{v}\|_2 = 1$ ,  $\max_{\mathbf{v} \in \mathcal{B}_d(1)} \mathbf{v}^\top \mathbf{X} =$

$\|\mathbf{X}\|$ . Consequently,

$$\begin{aligned} \Pr(\|\mathbf{Z} - \boldsymbol{\mu}\| \geq t) &\leq \Pr(\max_{\mathbf{v} \in \mathcal{B}_d(1)} \mathbf{v}^\top (\mathbf{W} - \boldsymbol{\mu}) \geq t) \\ &\leq \Pr(\max_{\mathbf{z} \in \mathcal{N}} \mathbf{z}^\top (\mathbf{W} - \boldsymbol{\mu}) \geq t/2) \\ &\leq 2|\mathcal{N}| \exp\left(-\frac{nt^2}{8\sigma^2}\right). \end{aligned}$$

Plugging in the value of  $|\mathcal{N}|$  yields the final result.  $\square$

**Lemma D.1.2.** *For any epoch  $k$  in PLS, the estimate  $\hat{\theta}_k^{(\text{SERV})}$  satisfies*

$$\|\hat{\theta}_k^{(\text{SERV})} - \theta^*\| \leq \tau_k,$$

*with probability at least  $1 - \delta$ .*

*Proof.* We prove the claim using induction. Firstly, note that if we define  $\bar{\theta}_{k-1} := 0$  for all epochs  $k$  during the norm estimation stage, then we can rewrite the method to compute the estimate at the server,  $\hat{\theta}_k^{(\text{SERV})}$ , during the norm estimation stage in the same way as it is computed during the refinement stage. Thus, we consider the definition of  $\hat{\theta}_k^{(\text{SERV})}$  as used in Algorithm 9 while implicitly assuming  $\bar{\theta}_{k-1} = 0$  for all epochs  $k$  during the norm estimation stage.

Let  $\eta_k^{(j)} \in \mathbb{R}^d$  denote the quantization noise added by  $j^{\text{th}}$  client during the  $k^{\text{th}}$  epoch, i.e., the quantized version received by the server can be written as  $Q(\tilde{\theta}_k^{(j)}) = \tilde{\theta}_k^{(j)} + \eta_k^{(j)}$ . Since each coordinate is quantized independently, each coordinate of  $\eta_k^{(j)}$  is an independent zero mean sub-Gaussian random variable with

variance proxy  $\alpha_k^2/4d$ . At the end of the  $k^{\text{th}}$  epoch, we have

$$\begin{aligned}
\|\hat{\theta}_k^{(\text{SERV})} - \theta^*\| &= \left\| \bar{\theta}_{k-1} + \frac{1}{M} \sum_{j=1}^M Q(\tilde{\theta}_k^{(j)}) - \theta^* \right\| \\
&= \left\| \bar{\theta}_{k-1} + \frac{1}{M} \sum_{j=1}^M (\tilde{\theta}_k^{(j)} + \eta_k^{(j)}) - \theta^* \right\| \\
&= \left\| \bar{\theta}_{k-1} + \frac{1}{M} \sum_{j=1}^M (\hat{\theta}_k^{(j)} - \bar{\theta}_{k-1}) \mathbb{1}_{R_k+B_k} + \frac{1}{M} \sum_{j=1}^M \eta_k^{(j)} - \theta^* \right\| \\
&\leq \left\| \frac{1}{M} \sum_{j=1}^M (\hat{\theta}_k^{(j)} - \bar{\theta}_{k-1}) \mathbb{1}_{R_k+B_k} - (\theta^* - \bar{\theta}_{k-1}) \right\| + \left\| \frac{1}{M} \sum_{j=1}^M \eta_k^{(j)} \right\|.
\end{aligned}$$

The second term can be bounded using the concentration of sub-Gaussian random vectors as

$$\left\| \frac{1}{M} \sum_{j=1}^M \eta_k^{(j)} \right\| \leq \frac{2\alpha_k}{\sqrt{M}} \left( 1 + \sqrt{\frac{1}{2d} \log \left( \frac{2K}{\delta} \right)} \right).$$

For the first term, we will employ Lemma D.1.1. However, we first need to ensure that the conditions of the lemma are satisfied. For any epoch  $k$ , the particular choice of  $R_k$  and  $s_k$  in PLS satisfies the assumption in Lemma D.1.1 for all  $k$ . To bound the mean, we need to consider the epochs with  $\bar{\theta}_{k-1} = 0$  (i.e., the norm estimation stage) and  $\bar{\theta}_{k-1} \neq 0$  (the refinement stage) separately. We begin with the case of  $\bar{\theta}_{k-1} = 0$ . Under this scenario, note that  $\|\mathbb{E}[\hat{\theta}_k^{(j)} - \bar{\theta}_{k-1}]\| = \|\mathbb{E}[\hat{\theta}_k^{(j)}]\| = \|\theta^*\|$ . For the first epoch, we know that  $\|\theta^*\| \leq 1 = B_1$ . This implies, we can apply the lemma for the base case. For any epoch  $k \geq 2$ , we use the inductive hypothesis to establish the bound on the mean. In particular, we use a better estimate of  $\|\theta^*\|$  by noting that the norm estimation stage had not been terminated by epoch  $k-1$  which implies that  $\|\hat{\theta}_{k-1}^{(\text{SERV})}\| \leq 4\tau_{k-1}$ . Along with the induction hypothesis,  $\|\hat{\theta}_{k-1}^{(\text{SERV})} - \theta^*\| \leq \tau_{k-1}$ , we can conclude that  $\|\theta^*\| \leq 5\tau_{k-1} \leq B_k$ , as required.

Now we are ready to invoke Lemma D.1.1. Using Lemma D.1.1, we can

conclude that with probability at least  $1 - \delta/K$ ,

$$\left\| \frac{1}{M} \sum_{j=1}^M (\hat{\theta}_k^{(j)} - \bar{\theta}_{k-1}) \mathbb{1}_{R_k + B_k} - (\theta^* - \bar{\theta}_{k-1}) \right\| \leq \frac{4\sigma \sqrt{d}}{\sqrt{Ms_k}} \left( 1 + \sqrt{\frac{1}{2d} \log \left( \frac{2K}{\delta} \right)} \right).$$

On plugging in the value of  $s_k$  and  $\alpha_k$  and combining the equations, we obtain,

$$\|\hat{\theta}_k^{(\text{SERV})} - \theta^*\| \leq \frac{2^{-k}}{\sqrt{M}} + \frac{2^{-(k+1)}}{\sqrt{M}} = \frac{3}{\sqrt{M}} \cdot 2^{-(k+1)} = \tau_k,$$

holds with probability at least  $1 - \delta/K$ .

We similarly consider the case where in epoch  $k$ ,  $\bar{\theta}_{k-1} \neq 0$ . For this case, we apply Lemma D.1.1 conditioned on the value of  $\bar{\theta}_{k-1}$  and hence all expectations and probabilities are computed with respect to the conditional measure. For brevity of notation and ease of understanding, we will assume the conditioning has been implicitly applied. The condition on  $R_k$  holds using the argument as in the previous case and hence we focus only on showing the bound on the mean. If  $\zeta_{k-1}$  denotes the quantization error during the  $k-1$  epoch, then we can write  $\bar{\theta}_{k-1} = \hat{\theta}_{k-1}^{(\text{SERV})} + \zeta_{k-1}$ . Consequently,  $\|\mathbb{E}[\hat{\theta}_k^{(j)} - \bar{\theta}_{k-1}]\| = \|\theta^* - \bar{\theta}_{k-1}\| = \|\theta^* - \hat{\theta}_{k-1}^{(\text{SERV})} - \zeta_{k-1}\| \leq \|\theta^* - \hat{\theta}_{k-1}^{(\text{SERV})}\| + \|\zeta_{k-1}\| \leq \|\theta^* - \hat{\theta}_{k-1}^{(\text{SERV})}\| + \beta_{k-1} \leq \tau_{k-1} + \beta_{k-1} \leq B_k$ . For the last step, the bound on  $\|\theta^* - \hat{\theta}_{k-1}^{(\text{SERV})}\|$  holds due to the hypothesis in the induction step. In the base case, i.e.,  $k_0$ , the index of the epoch when the norm estimation stage terminates, the bound holds from the result obtained for the case of  $\bar{\theta}_{k-1} = 0$ . This implies that we can apply the lemma also for the case of  $\bar{\theta}_{k-1} \neq 0$ , thereby obtaining the bound on  $\|\hat{\theta}_k^{(\text{SERV})} - \theta^*\|$  for all  $k$ .

We would like to point that to avoid repeating steps in the proof, we have directly shown the result after invoking Lemma D.1.1 for all epochs  $k$ . The proof actually follows by first invoking Lemma D.1.1 for the base case and establish-

ing the result. For any epoch  $k$ , we then use the inductive hypothesis of the relation for  $k - 1$  to establish the conditions on the mean after which Lemma D.1.1 is invoked to complete the induction step.

Lastly, consider the events given by

$$\begin{aligned}\mathcal{E}_k &:= \{\|\hat{\theta}_k^{(\text{SERV})} - \theta^*\| \leq \tau_k\} \quad \forall k \text{ in } \mathbb{N} \\ \mathcal{E} &:= \bigcap_{k=1}^K \mathcal{E}_k\end{aligned}\tag{D.1}$$

From the above analysis, we know that  $\Pr(\mathcal{E}_k | \mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_{k-1}) \geq 1 - \delta/K$  for  $k = 1, 2, \dots, K$ , where  $\Pr(\mathcal{E}_0) = 1$ . Thus, we have

$$\begin{aligned}\Pr(\mathcal{E}) &= \Pr\left(\bigcap_{k=1}^K \mathcal{E}_k\right) \\ &= \prod_{i=1}^k \Pr(\mathcal{E}_k | \mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_{k-1}) \\ &\geq \prod_{i=1}^K \left(1 - \frac{\delta}{K}\right)^K \\ &\geq 1 - \delta,\end{aligned}$$

as required. □

We now proceed to provide detailed proofs of the various theorems and lemmas that characterize the regret and communication cost of PLS.

### D.1.1 Proof of Lemma 5.4.2

Let  $R_{\text{NE}}(T)$  denote the regret incurred during the norm estimation stage. We assume that the stage continues until a `terminate` is received from the server

or each client has played a total of  $T$  actions, whichever happens first.

Under the event  $\mathcal{E}$  as defined in Eqn. (D.1), the algorithm will terminate at the end of epoch  $k_0$  where  $k_0 := \min\{k \in \mathbb{N} : 4\tau_k \leq \|\hat{\theta}_k^{(\text{SERV})}\|\}$ . We note that for all  $k$  for which the inequality  $4\tau_k \leq \|\hat{\theta}_k^{(\text{SERV})}\|$  holds, we also have the relation

$$\begin{aligned} \|\hat{\theta}_k^{(\text{SERV})} - \theta^*\| &\leq \tau_k \leq \frac{1}{4} \cdot \|\hat{\theta}_k^{(\text{SERV})}\| \\ \implies \|\theta^*\| &\in \left[ \frac{3}{4} \|\hat{\theta}_k^{(\text{SERV})}\|, \frac{5}{4} \|\hat{\theta}_k^{(\text{SERV})}\| \right] \\ \implies \|\hat{\theta}_k^{(\text{SERV})}\| &\in \left[ \frac{4}{5} \|\theta^*\|, \frac{4}{3} \|\theta^*\| \right]. \end{aligned}$$

Consequently, we also have  $\tau_k \leq \frac{1}{3} \cdot \|\theta^*\|$ . On plugging in the value of  $\tau_k$ , we obtain

$$\tau_k \leq \frac{1}{3} \cdot \|\theta^*\| \implies 2^{-k} \leq \frac{2\sqrt{M}}{9} \|\theta^*\| \implies k \geq \log_2 \left( \frac{9}{2\|\theta^*\| \sqrt{M}} \right).$$

Since  $k_0$  is the smallest natural number satisfying this relation,  $k_0 = \max \left\{ \left\lceil \log_2 \left( \frac{9}{2\|\theta^*\| \sqrt{M}} \right) \right\rceil, 1 \right\}$ .

We know that  $a_t, a^* \in \mathcal{A} = \{x : \|x\|_2 \leq 1\}$  for all  $t \in \{1, 2, 3, \dots, T\}$ . Hence, the instantaneous regret at time instant can be bounded as

$$r_t = \langle \theta^*, a^* \rangle - \langle \theta^*, a_t \rangle \leq \|\theta^*\|_2 \|a^* - a_t\|_2 \leq 2\|\theta^*\|_2.$$

Consequently, we can bound  $R_{\text{NE}}(T)$  as follows. If  $k_0 = 1$ , then  $R_{\text{NE}}(T)$  can be

simply upper bounded as  $Mds_1 = O(1)$ . If  $k_0 \geq 2$ , we have

$$\begin{aligned}
R_{\text{NE}}(T) &\leq \sum_{k=1}^{k_0} 2M\|\theta^*\|_2 \cdot ds_k \\
&\leq 2Md\|\theta^*\|_2 \cdot s_{k_0} \cdot k_0 \\
&\leq 2Md\|\theta^*\|_2 \cdot \left(32d\sigma^2 \log\left(\frac{8MK}{\delta}\right) 4^{\log_2(9/2\|\theta^*\|_2)+1} + 1\right) \cdot \left(\log_2\left(\frac{9}{2\|\theta^*\|_2}\right) + 1\right) \\
&\leq 2Md\|\theta^*\|_2 \cdot \left(2592 \frac{d\sigma^2}{\|\theta^*\|_2^2} \log\left(\frac{8MK}{\delta}\right) + 1\right) \cdot \left(\log_2\left(\frac{9}{2\|\theta^*\|_2}\right) + 1\right) \\
&\leq 5184 \frac{d^2\sigma^2}{\|\theta^*\|_2} \log\left(\frac{8MK}{\delta}\right) \left(\log_2\left(\frac{9}{2\|\theta^*\|_2}\right) + 1\right) + 2d \left(\log_2\left(\frac{9}{2\|\theta^*\|_2}\right) + 1\right).
\end{aligned}$$

A trivial bound on  $R_{\text{NE}}(T)$  is  $2\|\theta^*\|_2 MT$ . Note that for larger values of  $\|\theta^*\|_2$ , the former bound is tighter. However, as  $\|\theta^*\|_2$  gets smaller, the latter bound becomes a stronger one. To obtain the optimal bound on  $R_{\text{NE}}(T)$ , we choose the minimum of the two. Thus,

$$R_{\text{NE}}(T) \leq \min \left\{ 5184 \frac{d^2\sigma^2}{\|\theta^*\|_2} \log\left(\frac{8MK}{\delta}\right) \left(\log_2\left(\frac{9}{2\|\theta^*\|_2}\right) + 1\right) + 2d \left(\log_2\left(\frac{9}{2\|\theta^*\|_2}\right) + 1\right), 2\|\theta^*\|_2 MT \right\}.$$

On setting  $\|\theta^*\|_2 = d/\sqrt{MT}$ , we obtain  $R_{\text{NE}}(T)$  is  $O(d\sqrt{MT} \cdot \log(MT) \cdot \log(\log T/\delta))$ , as required.

### D.1.2 Proof of Lemma 5.4.3

We are given access to estimate,  $\hat{\theta}$ , of the true vector,  $\theta$ , such that  $\|\hat{\theta} - \theta\|_2 \leq \tau \leq \|\theta\|_2$ . This implies that  $\|\hat{\theta}\|_2 \in [\|\theta\|_2 - \tau, \|\theta\|_2 + \tau]$ . Consider the relation,

$$\begin{aligned} \|\hat{\theta} - \theta\|_2^2 &\leq \tau^2 \\ \implies \|\hat{\theta}\|_2^2 + \|\theta^*\|_2^2 - 2\langle \hat{\theta}, \theta \rangle &\leq \tau^2 \\ \implies -\langle \hat{\theta}, \theta \rangle &\leq \frac{1}{2}(\tau^2 - \|\hat{\theta}\|_2^2 - \|\theta\|_2^2) \\ \implies \|\theta\| - \frac{1}{\|\hat{\theta}\|_2} \langle \hat{\theta}, \theta \rangle &\leq \frac{1}{2\|\hat{\theta}\|_2} (\tau^2 - \|\theta\|_2^2 - \|\theta\|_2^2 + 2\|\theta\|_2\|\theta\|_2) \\ \implies \|\theta\| - \frac{1}{\|\hat{\theta}\|_2} \langle \hat{\theta}, \theta \rangle &\leq \frac{1}{2\|\theta\|_2} (\tau^2 - (\|\theta\|_2 - \|\theta\|_2)^2). \end{aligned}$$

Note that the LHS in the last equation is an upper bound on the regret incurred by playing the action  $a = \hat{\theta}/\|\hat{\theta}\|_2$ . If we denote the regret incurred by playing the action  $a$  by  $\text{Reg}(a)$  and let  $\|\hat{\theta}\|_2 = \|\theta\|_2 + \nu$ , for some  $\nu \in [-\tau, \tau]$ , then

$$\text{Reg}(a) \leq \frac{\tau^2 - \nu^2}{2(\|\theta\|_2 + \nu)}.$$

To bound the above expression, we consider the function  $f(\nu) = \frac{\tau^2 - \nu^2}{(\|\theta\|_2 + \nu)}$ . Since  $f$  is rational function of two polynomials, it is differentiable. On differentiating  $f$ , we obtain  $f'(\nu) = -\frac{\nu^2 + 2\nu\|\theta\|_2 + \tau^2}{(\|\theta\|_2 + \nu)^2}$ . The solutions for  $f'(\nu) = 0$  are given by  $\nu_+ = -\|\theta\|_2 + \sqrt{\|\theta\|_2^2 - \tau^2}$  and  $\nu_- = -\|\theta\|_2 - \sqrt{\|\theta\|_2^2 - \tau^2}$ . By double differentiating  $f$ , we can verify that it is indeed maximized at  $\nu_+$  for  $\nu \in [-\tau, \tau]$ . Moreover, note that solutions for  $f'(\nu) = 0$  are real numbers only for  $\tau \leq \|\theta\|_2$ . This explains the need for the constraint on  $\tau$ . On setting  $\nu = \nu_+$  in the expression for  $\text{Reg}(a)$ , we

obtain the following bound.

$$\begin{aligned}
\text{Reg}(a) &\leq \frac{\tau^2 - (-\|\theta^*\|_2 + \sqrt{\|\theta\|_2^2 - \tau^2})^2}{2\sqrt{\|\theta\|_2 - \tau^2}} \\
&\leq \frac{\tau^2 - \|\theta\|_2^2 - \|\theta\|_2^2 + \tau^2 + 2\|\theta\|_2\sqrt{\|\theta\|_2^2 - \tau^2}}{2\sqrt{\|\theta\|_2 - \tau^2}} \\
&\leq \|\theta\|_2 - \sqrt{\|\theta\|_2^2 - \tau^2} \\
&\leq \frac{\tau^2}{\|\theta\|}.
\end{aligned}$$

### D.1.3 Proof of Theorem 5.4.1

Let  $R_{\text{REF}}(T)$  denote the regret incurred during the refinement stage. To obtain a bound on  $R_{\text{REF}}(T)$ , we consider an epoch  $k$  during the refinement stage. Similar to the norm estimation stage, we bound the regret incurred during the exploration sub-epoch as  $2\|\theta^*\|_2 \cdot Mds_k$ . Using Lemma 5.4.3 and the fact that  $\|\bar{\theta}_k - \theta^*\| \leq 2\tau_k$  (Ref. Lemma D.1.2), we can bound the regret during the exploitation sub-epoch as  $\frac{4\tau_k^2}{\|\theta^*\|_2} \cdot Mt_k$ . If  $R^{(k)}$  denotes the regret incurred during epoch  $k$ , then we have the following relation.

$$\begin{aligned}
R^{(k)} &\leq 2\|\theta^*\|_2 \cdot Mds_k + \frac{4M\tau_k^2}{\|\theta^*\|_2} \cdot (Ms_k^2\mu_0^2 + 1) \\
&\leq 2\|\theta^*\|_2 \cdot Mds_k + \frac{100(M\tau_k s_k \|\theta^*\|_2)^2}{9\|\theta^*\|_2} + \frac{4M\tau_k^2}{\|\theta^*\|_2} \\
&\leq 64\|\theta^*\|_2 Md^2\sigma^2 \log\left(\frac{8MK}{\delta}\right) 4^k + 2\|\theta^*\|_2 \cdot Md \\
&\quad + 8M\|\theta^*\|_2 \left(6400\sigma^4 d^2 \log^2\left(\frac{8MK}{\delta}\right) 4^k + 4^{-k}\right) + M\|\theta^*\|_2 \\
&\leq 51200 \cdot \|\theta^*\|_2 Md^2\sigma^2 (1 + \sigma^2) \log^2\left(\frac{8MK}{\delta}\right) 4^k + M\|\theta^*\|_2 (8 \cdot 4^{-k} + 2d + 1)
\end{aligned}$$

Note that the regret incurred during the exploration sub-epoch is the same as that incurred during the exploitation sub-epoch upto constant factors. This

echoes the discussion in Section 5.4 on the careful choice of the lengths of exploration and exploitation epochs in PLS using the estimated norm of  $\|\theta^*\|_2$ .

Let  $k_1$  denote the index of the epoch when the query budget ends. Also, recall that  $k_0$  was defined to be the epoch index during which the norm estimation stage terminates. If  $k_1 = k_0$ , then it implies that the exploitation sub-epoch of the  $k_0^{\text{th}}$  epoch was not completed. Consequently, the regret incurred during the partial sub-epoch is bounded by the regret incurred during the corresponding exploration sub-epoch (upto a constant factor) which in turn is bounded by the regret incurred during the norm estimation stage. Hence, the overall regret has the same order as the of the norm estimation stage, that is,  $\tilde{O}(d \sqrt{MT})$ , as required. For the rest of the proof we focus on the case  $k_1 \geq k_0 + 1$ .

The regret incurred during the refinement stage can be bounded as

$$R_{\text{REF}}(T) \leq \sum_{k=k_0}^{k_1} R^{(k)}.$$

Since we already have a bound on  $k_0$ , we can bound the above expression by finding an upper bound on  $k_1$ . We can bound  $k_1$  by using the length of the time horizon. We have,

$$\sum_{k=1}^{k_1-1} ds_k + \sum_{k=k_0}^{k_1-1} t_k \leq T.$$

The first term corresponds to the length of all the exploration (sub-)epochs, including the ones during the norm estimation procedure, while the second accounts for the length of all the exploitation sequences. On plugging in the values

of  $s_k$  and  $t_k$ , we obtain,

$$\begin{aligned}
T &\geq \sum_{k=1}^{k_1-1} ds_k + \sum_{k=k_0}^{k_1-1} t_k \\
&\geq \sum_{k=k_0}^{k_1-1} Ms_k^2 \mu_0^2 \\
&\geq \sum_{k=k_0}^{k_1-1} \frac{16384}{25} Md^2 \|\theta^*\|^2 \sigma^4 \log^2 \left( \frac{8MK}{\delta} \right) 16^k \\
&\geq \frac{1024}{9} Md^2 \|\theta^*\|^2 \sigma^4 \log^2 \left( \frac{8MK}{\delta} \right) \frac{16^{k_1} - 16^{k_0}}{15} \\
&\geq 100Md^2 \|\theta^*\|^2 \sigma^4 \log^2 \left( \frac{8MK}{\delta} \right) 16^{k_1}.
\end{aligned}$$

where the last step follows by noting that  $k_1 \geq k_0 + 1$ . This implies that  $k_1 \leq \log_{16} \left( \frac{T}{100Md^2 \|\theta^*\|_2^2 \sigma^4} \left( \log \left( \frac{8MK}{\delta} \right) \right)^{-2} \right)$ .

Consequently,

$$\begin{aligned}
R_{\text{REF}}(T) &\leq \sum_{k=k_0}^{k_1} R^{(k)} \\
&\leq \sum_{k=k_0}^{k_1} 51200 \cdot \|\theta^*\|_2 Md^2 \sigma^2 (1 + \sigma^2) \log^2 \left( \frac{8MK}{\delta} \right) 4^k + M \|\theta^*\|_2 (8 \cdot 4^{-k} + 2d + 1) \\
&\leq 69000 \cdot \|\theta^*\|_2 Md^2 \sigma^2 (1 + \sigma^2) \log^2 \left( \frac{8MK}{\delta} \right) 4^{k_1} + Mk_1 \|\theta^*\|_2 (2d + 9) \\
&\leq 69000 \cdot \|\theta^*\|_2 Md^2 \sigma^2 (1 + \sigma^2) \log^2 \left( \frac{8MK}{\delta} \right) \cdot \frac{1}{d \sigma^2 \|\theta^*\|_2} \left( \log \left( \frac{8MK}{\delta} \right) \right)^{-1} \cdot \sqrt{\frac{T}{100M}} \\
&\quad + Mk_1 \|\theta^*\|_2 (2d + 9) \\
&\leq 6900(1 + \sigma^2) \log \left( \frac{8MK}{\delta} \right) \cdot d \sqrt{MT} + Mk_1 \|\theta^*\|_2 (2d + 9).
\end{aligned}$$

Adding this to the regret incurred during the norm estimation stage, we can conclude that  $R_{\text{PLS}}(T)$  satisfies  $O(d \sqrt{MT} \cdot \log(MT) \cdot \log(\log T/\delta))$ .

#### D.1.4 Proof of Lemma 5.4.5

Consider a vector  $x \in \mathbb{R}^d$  with  $\|x\| \leq r$  and let  $Q(x)$  denote its quantized version achieved by a quantizer (deterministic or stochastic) upto a precision of  $\varepsilon$ . This implies that  $\|x - Q(x)\|_2 \leq \varepsilon \implies \|Q(x)\|_2 \leq r + \varepsilon$ . Note that  $Q(x)$  can be represented as  $(q_1, q_2, \dots, q_d)^\top$  where  $q_i \in \{-l_\varepsilon/2, \dots, 0, \dots, l_\varepsilon/2\}$  represents the corresponding quantized index for all  $i \in \{1, 2, \dots, d\}$ . Thus, we have,

$$\begin{aligned} \sum_{i=1}^d q_i^2 \left(\frac{2r}{l_\varepsilon}\right)^2 &\leq (r + \varepsilon)^2 \\ \implies \sum_{i=1}^d q_i^2 &\leq \left(\frac{(r + \varepsilon)l_\varepsilon}{2r}\right)^2 \\ &\leq 4d \left(\frac{r}{\varepsilon} + 1\right)^2. \end{aligned}$$

Consequently,

$$\sum_{i=1}^d |q_i| \leq \sqrt{d \sum_{i=1}^d q_i^2} \leq 2d \left(\frac{r}{\varepsilon} + 1\right).$$

It can be noted that under the encoding scheme based on unary encoding used in PLS, the message size in bits in PLS is given by  $3d + \sum_{i=1}^d |q_i|$ , where  $3d$  corresponds to the sum of lengths of the headers. As a result, the message size is bounded by  $d(3 + 2(r/\varepsilon + 1))$ . We use this relation to establish the message sizes on both the uplink and the downlink channels.

Let us first consider the uplink communication. In epoch  $k$ ,  $r$  corresponds to  $R_k + B_k$  and  $\varepsilon$  to  $\alpha_k$ . On plugging in the prescribed values of the above parameters, we note that  $(R_k + B_k)/\alpha_k$  is  $C/\alpha_0$ , where  $C$  is a constant independent of  $d, M$  and  $T$ . Hence, any message sent on the uplink channel is no more than  $O(d)$  bits. Similarly, for the downlink communication, the ratio  $(B_k + \tau_k)/\beta_k \leq C'/\beta_0$  where

once again  $C'$  is a constant independent of  $d, M$  and  $T$ . Hence, any message sent on the downlink channel also has a size of  $O(d)$  bits, as required.

### D.1.5 Proof of Theorem 5.4.4

The bound on the uplink communication cost follows immediately from Lemma 5.4.5. Since the agents send a message of  $O(d/\alpha_0)$  bits only once every epoch and there are no more than  $K = O(\log T)$  epochs in PLS, the uplink communication cost of PLS,  $C_u(T)$ , satisfies  $O((d/\alpha_0) \log T)$ . The  $O((d/\beta_0) \log T)$  term in  $C_d(T)$ , the downlink cost, also follows using the same argument.

The  $O(d \log M)$  term in the downlink communication cost corresponds to sending the initial estimate of  $\theta^*$  to the clients, specifically in the event that the norm estimation stage ends within the first epoch. Note that, if the norm estimation stage ends in the first epoch, the estimation error in  $\theta^*$  becomes  $\tau_1 = O(\sqrt{1/M})$  from the initial estimate of  $O(1)$ . As a result, PLS requires  $O(d \log M)$  bits to transmit the estimate, adding to the downlink communication cost. This  $O(d \log M)$  cost is what facilitates information exchange between the agents that leads to the linear speedup with respect to the number of agents.

*Remark D.1.3.* This estimate is also sent using the unary encoding scheme used in PLS over  $O(\log M)$  rounds with each round sending one additional bit of information per coordinate. Depending on the synchronization requirements as dictated by the hardware, the learner may not be allowed multiple uses of the channel. In the event that it is possible to use the channel several times between two actions, this information can be easily sent with  $O(\log M)$  uses of the channel. However, if the server is allowed to use the channel only once between two

actions of the agent, this transmission of the initial estimate can be accommodated within PLS as follows. At the end of the exploration sub-epoch of the first epoch, instead of starting with the exploitation sub-epoch, the agents begin with the exploration sub-epoch of the second epoch. During this time, the server broadcasts the initial estimate of  $\theta^*$  over the next  $O(\log M)$  time instants. The agents continue to play the actions as dictated by the exploration sub-epoch until the communication is completed. Once the communication is completed, the agents now start with the exploitation sub-epoch of the first epoch. At the end of the first epoch, they restart the exploration sub-epoch, starting from where they had left at the end of the communication. Thus, if needed, PLS can accommodate the additional transmission requirements by a minor reordering of the explorative and exploitative actions.

## D.2 Establishing the Lower Bound

In this section, we discuss the details of the proofs to establish the lower bounds on the communication cost under regret constraints.

### D.2.1 Proof of Theorem 5.5.1

The proof of this theorem consists of two main steps. In the first step, we show that all algorithms achieving a sub-linear cumulative regret need to solve the problem of distributed mean estimation. This reduction allows us to leverage several existing techniques and results for information-theoretic lower bounds, especially in the case of distributed statistical estimation. The second step is

to establish lower bound on communication cost (both uplink and downlink) for a given estimation error based on the above reduction. The primary idea for this step is to use to classical reduction to identification from a specifically constructed hard instance. Specifically, we show that for any estimator with a small error it is necessary to solve the identification problem. The final bound on the communication cost is then obtained by using Fano's inequality to bound the error of this identification problem.

We first establish how the constraint of a sub-linear cumulative regret for linear bandit algorithm can be translated to having a small simple regret. Consider any policy  $\pi$  that achieves a sub-linear regret  $R_\pi(T)$  under the setup described in Section 5.2. Consequently, the policy  $\pi$  also guarantees a sub-linear regret of  $R_\pi(T)/MT$ . This follows immediately by choosing the final action to be the average of all the actions chosen by all the agents. Note that this is a permissible action since  $\mathcal{A}$  is a convex set. As a result, a lower bound of  $\underline{r}(T)$  on simple regret achievable by any policy immediately implies a lower bound of  $\underline{R}(T) = M\underline{r}(T)$  on the cumulative regret achievable by any policy. This relation is used later in the proof to draw parallels with the problem of distributed mean estimation.

For the second step, we separately establish the lower bounds for the uplink and downlink communication cost, by constructing two different hard instances. We begin with the lower bound on the downlink cost. We would like to point out that we prove a more general case for the downlink cost where we allow algorithms that incur a sublinear w.r.t to both  $T$  and  $M$ , as opposed to just being order optimal.

## Proof of downlink cost

For the lower bound on the downlink cost, recall that for a policy  $\pi$  with sub-linear cumulative regret of  $R_\pi(T)$ , the average of all the actions chosen by all the agents, denoted by  $\bar{a}_\pi$ , achieves a simple regret of  $R_\pi(T)/MT$ . This implies that using the actions taken by  $\pi$ , one can estimate  $\theta^*$  to a reasonable accuracy dictated by  $R_\pi(T)$ . In particular, let  $\hat{\theta}(A; \pi)$  denote an estimator of  $\theta^*$  based on all the actions taken by a policy  $\pi$ , which we denote by the random variable  $A$ , and  $\|\theta^*\|_2 = 1$ . Then,

$$\begin{aligned} \inf_{\hat{\theta}} \|\hat{\theta}(A; \pi) - \theta^*\|_2^2 &\leq \left\| \frac{\bar{a}_\pi}{\|\bar{a}_\pi\|_2} - \theta^* \right\|_2^2 \\ &\leq \frac{\|\bar{a}_\pi\|_2^2}{\|\bar{a}_\pi\|_2^2} + \|\theta^*\|_2^2 - 2 \left\langle \theta^*, \frac{\bar{a}_\pi}{\|\bar{a}_\pi\|_2} \right\rangle \\ &= 2(1 - \langle \theta^*, \bar{a}_\pi \rangle) \\ &\leq 2 \frac{R_\pi(T)}{MT}. \end{aligned}$$

We use the above relation to obtain a bound on the downlink communication cost based on the information carried in  $A$ , the actions taken by a policy, about the underlying vector  $\theta^*$ .

Let  $\mathcal{V}_\varepsilon$  denote a maximal  $2\varepsilon$ -packing of  $\mathcal{S}^d$  using the  $\|\cdot\|_2$  norm, where  $\mathcal{S}^d$  denotes the surface of a unit sphere in  $\mathbb{R}^d$ . Equivalently,  $\mathcal{S}^d = \partial \mathcal{B}_d(1) \in \mathbb{R}^{d-1}$ , where  $\mathcal{B}_d(r)$  denotes a ball (in  $\|\cdot\|_2$ -norm) of radius  $r$  in  $\mathbb{R}^d$ . Let  $V$  be a random variable chosen uniformly at random from  $\mathcal{V}_\varepsilon$ . Consider a linear bandit instance instantiated with  $\theta^* = V$ . Note that this construction implicitly ensures  $\|\theta^*\|_2 = 1$ . As before, let  $\hat{\theta}(A; \pi)$  denote an estimator of  $\theta^*$  based on all the actions taken by a policy  $\pi$ . This problem of estimating  $\theta^*$  can be mapped to that of identifying  $V$  using classical techniques by defining a testing function  $\hat{V}$  for each estimator  $\hat{\theta}$ . In particular, define  $\hat{V} := \arg \min_{v \in \mathcal{V}_\varepsilon} \|\hat{\theta}(A; \pi) - v\|_2$ , that is, it maps  $\hat{\theta}$  to the closest

point in the set  $\mathcal{V}_\varepsilon$ .

Since  $\mathcal{V}_\varepsilon$  is  $2\varepsilon$ -packing,  $\|\hat{\theta}(A; \pi) - V\|_2 > \varepsilon$  whenever  $\hat{V} \neq V$ . Consequently,  $\Pr(\|\hat{\theta}(A; \pi) - \theta^*\|_2^2 > \varepsilon^2/2) = \Pr(\|\hat{\theta}(A; \pi) - V\|_2^2 > \varepsilon^2/2) \geq \Pr(\hat{V} \neq V)$ . Since  $V \rightarrow A \rightarrow \hat{V}$  from a Markov chain, using Fano's inequality [74], we can conclude that

$$\Pr(\hat{V} \neq V) \geq 1 - \frac{I(V; A) + \log 2}{|\mathcal{V}_\varepsilon|},$$

where  $I(V; A)$  denotes the mutual information between  $V$  and  $A$ . Note that the event  $\{\|\hat{\theta}(A; \pi) - \theta^*\|_2^2 > \varepsilon^2/2\}$  implies that no estimator based on the actions of  $\pi$  can estimate  $\theta^*$  within an error of  $\varepsilon^2/2$  implying that  $\pi$  incurs a cumulative regret of at least  $\varepsilon^2 MT/4$ . Hence, to ensure a cumulative regret of  $R_\pi(T)$  with probability at least  $2/3$ , we need to ensure  $\Pr(\hat{V} \neq V) \leq \Pr(\|\hat{\theta}(A; \pi) - \theta^*\|_2^2 > \varepsilon^2/2) < 1/3$  for  $\varepsilon = 2\sqrt{R_\pi(T)/MT}$ . Equivalently,  $I(V; A) \geq 2\log|\mathcal{V}_\varepsilon|/3 - \log 2$  with  $\varepsilon = 2\sqrt{R_\pi(T)/MT}$ . Using the standard bounds on packing numbers [24] and noting that  $R_\pi(T)$  is sub-linear in both  $M$  and  $T$ , we can conclude that  $I(V; A) \geq \Omega(d\log(MT))$ . The bound on the communication cost is obtained using the data processing inequality. Let  $Z$  denote all the messages broadcast by the server. Then  $I(V; Z)$  is a lower bound on the downlink communication cost. Notice that  $Z$  obeys the Markov chain  $V \rightarrow Z \rightarrow A \rightarrow \hat{V}$  since the actions taken by the agents change with  $V$  through the messages broadcast by the server.

From data processing inequality, we have,  $I(V; Z) \geq I(V; A)$ . In other words, since the messages  $Z$  transfer the information about the actions of other agents to any given agent, they should at least have as much information about  $\theta^*$  (or equivalently  $V$ ) as much as  $A$ , the set of all actions, does. Combining this with the bound on  $I(V; A)$ , we arrive the required lower bound on the downlink communication cost.

## Proof for uplink cost

We establish bounds on the uplink cost by drawing parallels to the distributed mean estimation problem. Consider the problem of distributed mean estimation where  $X$  denotes the random variable corresponding to the observations by the agents and  $Y$  denotes the messages sent by the agents to the server. Based on the messages  $Y$ , if no estimator can recover  $\theta^*$  to within a mean squared error of  $\varepsilon^2$ , then no linear bandit algorithm can achieve a simple regret of  $\varepsilon^2$ . This is similar to the argument shown for the downlink case. This implies that, only if  $\theta^*$  can be estimated to within an accuracy of  $\varepsilon^2$ , can there exist a linear bandit algorithm with a simple regret  $\varepsilon^2$  and hence potentially with a cumulative regret of  $2\varepsilon^2 MT$ . Thus, we consider the problem of distributed mean estimation and show that unless all agents send  $\Omega(d \log T)$  bits of information to the server, no estimator can estimate  $\theta^*$  within an accuracy of  $1/MT$  with probability at least  $2/3$ . For this case, we only consider the optimal rates instead of any sublinear rates.

The proof borrows ideas and techniques from the classical results in Duchi *et al.* [103]. In particular, the proof is similar to those of Proposition 2 and Theorem 1 in [103]. However, it is different in its own sense as it builds upon those results and strengthens the lower bounds with the missing logarithmic factors. Thus, this proof might be of independent interest to the larger community.

The basic idea in the proof is to construct a Markov chain  $\mathbf{V} \rightarrow X \rightarrow Y$ , where  $X$  represents the collection of all the observations at all the agents and  $Y$  denotes the messages sent by the agents. In particular,  $X = (X^{(1)}, X^{(2)}, \dots, X^{(M)})$  and  $Y = (Y_1, Y_2, \dots, Y_M)$ , where  $X^{(j)}$  denotes the set of observations at agent  $j$  and  $Y_j$  denotes the messages sent by agent  $j$  to the server. Since the agents cannot

communicate with each other directly, we assume  $Y_j$  depends only on  $X^{(j)}$ .

We begin with construction the hard instance for the random variable  $\mathbf{V}$ . Let  $\mathcal{V} = \{\pm 1\}^{d-1}$  and  $\mathbf{v} = (v_1, v_2, \dots, v_\ell) \in \mathcal{V}^\ell$  for some  $\ell \in \mathbb{N}$  specified later. For each  $\mathbf{v} \in \mathcal{V}^\ell$ , we define a vector  $\mu_{\mathbf{v}} \in R^{d-1}$  given by

$$\mu_{\mathbf{v}} := \sum_{r=1}^{\ell} \frac{2^{-r}}{\sqrt{d-1}} v_r.$$

Note that  $\|\mu_{\mathbf{v}}\|_2 \leq \sum_{r=1}^{\ell} \frac{2^{-r}}{\sqrt{d-1}} \|v_r\| \leq \sum_{r=1}^{\ell} 2^{-r} \leq 1$ . Thus,  $\mu_{\mathbf{v}} \in \mathcal{B}_{d-1}(1)$  for all  $\mathbf{v} \in \mathcal{V}^\ell$ . We also define a lifting map  $f : \mathcal{B}_{d-1}(1) \rightarrow \mathcal{S}_{\geq 0}^d$ , where  $\mathcal{S}_{\geq 0}^d$  denotes the hemisphere where the last coordinate only takes on non-negative values. It maps a vector  $x \in \mathcal{B}_{d-1}(1)$  to a vector  $x' \in \mathcal{S}_{\geq 0}^d$ , where  $x'_{1:d-1} = x$  and  $x'_d = \sqrt{1 - \|x\|^2}$ . In other words,  $f$  lifts a point in unit hypersphere in  $d-1$  to the corresponding point on the unit hyper(hemi)sphere in  $\mathbb{R}^d$  by adding the last component. From the definition, one can note that  $f$  is a bijection. Furthermore, one can note that for any two points  $x, x' \in \mathcal{B}_{d-1}$ , we have  $\|x - x'\|_2 \leq \|f(x) - f(x')\|_2 \leq \sqrt{2}\|x - x'\|_2$ .

Let  $\mathbf{V}$  be a random variable drawn from the set  $\mathcal{V}^\ell$ . We construct a linear bandit instance with  $\theta^* = f(\delta\mu_{\mathbf{v}})$  for some  $\delta \in (0, 1)$  whose value is specified later. Since,  $f$  is a bijection that preserves distances upto a constant, it equivalent to consider the problem of estimating  $\theta^* = \theta_{\mathbf{v}} = \delta\mu_{\mathbf{v}}$ , where the operation  $f$  is assumed to have been carried out implicitly. Hence, for the remainder of the proof, we focus only of the problem on estimating  $\theta_{\mathbf{v}}$ .

To specify the observation model, we first need to set up some notations and definitions. Given a  $u \in \{-1, 1\}$  and  $\rho \in [0, 1]$ , we define  $P_{u,\rho}$  to be the distribution that assigns a mass of  $(1 + \rho)/2$  to  $u$  and  $(1 - \rho)/2$  to  $-u$ . We overload the definition of  $P_{u,\rho}$  for when  $u = (u_1, u_2, \dots, u_p)^\top \in \{-1, 1\}^p$  is a vector. In this case, a sample from  $P_{u,\rho}$  is a vector whose  $i^{\text{th}}$  coordinate is drawn according to

$P_{u,\rho}$ , independently of others. For a given value of  $\mathbf{V} = \boldsymbol{\nu}$ , each agent  $j$ , independent of other agents, receives an collection of  $T$  i.i.d. samples, denoted by  $X^{(j)} = \{X^{(j,k)}\}_{k=1}^n$  for  $j = 1, 2, \dots, M$ . Each sample  $X^{(j,k)}$  is obtained by first drawing  $\tilde{V}^{(j,k)}(\boldsymbol{\nu}) = (\tilde{V}^{(j,k,1)}(\boldsymbol{\nu}), \dots, \tilde{V}^{(j,k,\ell)}(\boldsymbol{\nu})) \in \mathcal{V}^\ell$ , where  $\tilde{V}^{(j,k,r)}(\boldsymbol{\nu}) \sim P_{v_r, \rho_0}$  for  $r = 1, 2, \dots, \ell$  and then setting  $X^{(j,k)} = \mu_{\tilde{V}^{(j,k)}(\boldsymbol{\nu})}$ . In the above definition  $\rho_0 := \delta/(T\ell)$ . If  $X_i^{(j)}$  denotes the vector obtained by taking the  $i^{\text{th}}$  coordinate of all the  $T$  samples at agent  $j$ , then from the above definition, one can note that  $X_i^{(j)}$ 's are independent across  $i$ , that is, each coordinate is independent of others.

Having specified the underlying mean vector and the observation model, the next step is to use strong data processing inequality [103] to quantitatively relate  $I(\mathbf{V}; Y_j)$  and  $I(X^{(j)}; Y_j)$ . To establish this relation, we make use of Lemma 5 from [103]. Before invoking the Lemma, we first need to establish a bound on the likelihood ratio of any realization  $x_i$  of the random variable  $X_i^{(j)}$ , under two different values of  $\mathbf{V}$ , say  $\boldsymbol{\nu}, \boldsymbol{\nu}'$  and also specify the measurable sets over which the bound holds. Throughout this part, we carry out all the analysis for a fixed agent  $j$ .

Note that any realization  $x_i$  can be mapped to the realizations  $\{\tilde{v}_i^{(j,k)}\}_{k=1}^T$ , where  $\tilde{v}_i^{(j,k)} = (\tilde{v}_i^{(j,k,1)}, \dots, \tilde{v}_i^{(j,k,\ell)})$ . For any fixed value of  $r \in \{1, 2, \dots, \ell\}$ , consider the set  $\tilde{v}_i^{(j,1:T,r)} = \{\tilde{v}_i^{(j,1:T,r)}\}_{k=1}^T$  which only contains values from  $\{-1, 1\}$  drawn from  $P_{v_{r,i}, \rho_0}$ . Here  $v_{r,i}$  denotes the  $i^{\text{th}}$  coordinate of  $v_r$ . Suppose this collection contains  $T_1$  instances of  $+1$  and  $T - T_1$  of  $-1$ , then the likelihood ratio any under any pair  $(\boldsymbol{\nu}, \boldsymbol{\nu}')$  can be bounded as  $\left(\frac{1+\rho_0}{1-\rho_0}\right)^{|T-2T_1|}$ . If  $S_r$  denotes the absolute value of sum of the elements in  $\tilde{v}_i^{(j,1:T,r)}$ , then this bound on the likelihood ratio can be rewritten as  $\left(\frac{1+\rho_0}{1-\rho_0}\right)^{S_r}$ . Since all  $\ell$  sequences in  $\{\tilde{v}_i^{(j,1:T,r)}\}_{r=1}^\ell$  are independent of each other, the

likelihood ratio can be bounded as

$$\sup_{x_i} \sup_{\mathbf{v}, \mathbf{v}' \in \mathcal{V}^\ell} \frac{\Pr(x_i | \mathbf{v})}{\Pr(x_i | \mathbf{v}') \leq \left( \frac{1 + \delta/(T\ell)}{1 - \delta/(T\ell)} \right)^{S_1} \cdots \left( \frac{1 + \delta/(T\ell)}{1 - \delta/(T\ell)} \right)^{S_\ell} = \left( \frac{1 + \delta/(T\ell)}{1 - \delta/(T\ell)} \right)^{S_1 + \cdots + S_\ell}.$$

To construct a measurable set  $G_i$  corresponding to each coordinate  $i$  on which the likelihood ratio can be appropriately bounded, consider the set  $\mathcal{Z}$  defined as

$$\mathcal{Z} = \left\{ (v_1, \dots, v_T) \in \{-1, +1\}^T : \left| \sum_{r=1}^T v_r \right| \leq a \sqrt{2T} + \delta/\ell \right\}$$

for some  $a > 0$  to be determined later. We set  $G_i = \{x_i : \tilde{v}_i^{(j,1:T,r)} \in \mathcal{Z} \forall r \in \{1, 2, \dots, \ell\}\}$  for all coordinates  $i$ . That is,  $G_i$  consists of all sequences  $x_i$  such that all its corresponding  $\tilde{v}$  sequences belong to  $\mathcal{Z}$ . When  $X_i^{(j)}$  belongs to the  $\sigma$ -field generated by the elements of  $G_i$ , we can bound the likelihood ratio as

$$\begin{aligned} \sup_{x_i \in \sigma(G_i)} \sup_{\mathbf{v}, \mathbf{v}' \in \mathcal{V}^\ell} \frac{\Pr(x_i | \mathbf{v})}{\Pr(x_i | \mathbf{v}')} &\leq \left( \frac{1 + \delta/(T\ell)}{1 - \delta/(T\ell)} \right)^{\ell(a \sqrt{2T} + \delta/\ell)} \\ &\leq \exp \left( 3\ell(a \sqrt{2T} + \delta/\ell) \cdot \frac{\delta}{(T\ell)} \right) \\ &\leq \exp \left( 3(a + \delta/\ell)\delta \sqrt{\frac{2}{T}} \right) := \exp(\varphi). \end{aligned}$$

for all  $T \geq 4\ell/3$ . As the last step to invoke the Lemma, we define  $E_i^{(j,r)} = \mathbb{1}\{\tilde{v}_i^{(j,1:T,r)} \in \mathcal{Z}\}$  and  $E_i^{(j)} = \prod_{r=1}^\ell E_i^{(j,r)}$ , where  $\mathbb{1}\{A\}$  denotes the indicator variable for the event  $A$ . Using the definition of  $G_i$ , we can also define  $E_i^{(j)}$  as  $\mathbb{1}\{X_i^{(j)} \in G_i\}$ . Lastly, we also define  $E^{(j)} = \prod_{i=1}^{d-1} E_i^{(j)}$ .

We are now all set to invoke Lemma 5 from [103]. Using the Lemma, we can conclude that for the message sent by agent  $j$ ,  $Y_j$ , the following inequality holds

$$I(\mathbf{V}; Y_j) \leq 2(e^{4\varphi} - 1)^2 I(X^{(j)}; Y_j | E^{(j)} = 1) + \sum_{i=1}^{d-1} H(E_i^{(j)}) + \sum_{i=1}^{d-1} H(\mathbf{V}_i) \Pr(E_i^{(j)} = 0),$$

where  $H(W)$  denotes the entropy of the random variable  $W$  and  $\mathbf{V}_i$  refers to the tuple formed by taking the  $i^{\text{th}}$  coordinate of all the elements in tuple represented by  $\mathbf{V}$ .

For the first term, note that for  $T \geq 200(a+1)^2$ ,  $\varphi \leq 5/16$  implying that  $2(\exp(4\varphi) - 1)^2 \leq 2(8\varphi)^2 = 128\varphi^2$ . Using the standard concentration bounds for Bernoulli random variables, we can conclude that  $\Pr(E_i^{(j,r)} = 0) \leq 2\exp(-a^2)$  and consequently  $\Pr(E_i^{(j)} = 0) \leq 2\ell\exp(-a^2)$ . On plugging these results into the previous equation, we obtain,

$$\begin{aligned}
I(\mathbf{V}; Y_j) &\leq \frac{2304(a + \delta/\ell)^2 \delta^2}{T} I(X^{(j)}; Y_j | E^{(j)} = 1) + \sum_{i=1}^{d-1} \sum_{r=1}^{\ell} H(E_i^{(j,r)}) + 2 \sum_{i=1}^{d-1} \ell^2 \exp(-a^2) \\
&\leq \frac{2304(a + \delta/\ell)^2 \delta^2}{T} \sum_{k=1}^T I(X^{(j,k)}; Y_j | E^{(j)} = 1, X^{(j,1:(k-1))}) + (d-1)\ell(h_2(2e^{-a^2}) + 2\ell e^{-a^2}) \\
&\leq \frac{2304(a + \delta/\ell)^2 \delta^2}{T} \sum_{k=1}^T \min\{H(X^{(j,k)} | E^{(j)} = 1, X^{(j,1:(k-1))}), H(Y_j | E^{(j)} = 1, X^{(j,1:(k-1))})\} \\
&\quad + (d-1)\ell(h_2(2e^{-a^2}) + 2\ell e^{-a^2}) \\
&\leq \frac{2304(a + \delta/\ell)^2 \delta^2}{T} \sum_{k=1}^T \min\{H(X^{(j,k)}), H(Y_j)\} + (d-1)\ell(h_2(2e^{-a^2}) + 2\ell e^{-a^2}) \\
&\leq \frac{2304(a + \delta/\ell)^2 \delta^2}{T} \sum_{k=1}^T \min\{(d-1)\ell, H(Y_j)\} + (d-1)\ell(h_2(2e^{-a^2}) + 2\ell e^{-a^2}) \\
&\leq 2304(a + \delta/\ell)^2 \delta^2 \min\{(d-1)\ell, H(Y_j)\} + (d-1)\ell(h_2(2e^{-a^2}) + 2\ell e^{-a^2}),
\end{aligned}$$

where we used the relation  $I(W; W') \leq \min\{H(W), H(W')\}$  for two random variables along with the fact that conditioning reduces entropy. In the above equations  $h_2(p) := -p \log_2(p) - (1-p) \log_2(1-p)$ , denotes the entropy of a Bernoulli random variable with mean  $p$ , for  $p \in [0, 1]$ . Since each message  $Y_j$  depends

only on  $X^{(j)}$ , we have,

$$\begin{aligned} I(\mathbf{V}; Y) &\leq \sum_{j=1}^M I(\mathbf{V}; Y_j) \\ &\leq \sum_{j=1}^M \left[ 2304(a + \delta/\ell)^2 \delta^2 \min\{(d-1)\ell, H(Y_j)\} + (d-1)\ell(h_2(e^{-a^2}) + \ell e^{-a^2}) \right]. \end{aligned}$$

This gives us the bound on  $I(\mathbf{V}; Y)$  in terms of  $H(Y_j)$ , which is a lower bound on the required communication cost to send the message corresponding to agent  $j$ . The last step is use Fano's inequality to translate this bound to a bound on the estimation error.

Let  $\hat{\theta}(Y)$  denote the estimate obtained at the server using the received messages  $Y$ . Similar to the previous case, let  $\hat{\mathbf{V}} := \arg \min_{\mathbf{v} \in \mathcal{V}'} \|\hat{\theta}(Y) - \theta_{\mathbf{v}}\|_2$  denote the corresponding estimate of  $\mathbf{V}$ . For  $\hat{\mathbf{V}} \neq \mathbf{V}$ , the estimation error  $\|\hat{\theta}(Y) - \theta_{\mathbf{v}}\|_2^2$  satisfies  $\|\hat{\theta}(Y) - \theta_{\mathbf{v}}\|_2^2 \geq \|\theta_{\hat{\mathbf{V}}} - \theta_{\mathbf{V}}\|_2^2 / 4 \geq \delta^2 \|\mu_{\hat{\mathbf{V}}} - \mu_{\mathbf{V}}\|_2^2 \geq \delta^2 4^{-\ell-1}/(d-1)$ . The last bound follows by noting that  $\|\mu_{\mathbf{v}} - \mu_{\mathbf{v}'}\| \geq \delta 2^{-\ell} / \sqrt{d-1}$  for  $\mathbf{v} \neq \mathbf{v}'$ .

We set  $a := 2\sqrt{\log(4M\ell)}$ ,  $\delta^2 := \left[ 9216(a+1/\ell) \sum_{j=1}^M \min\left\{1, \frac{H(Y_j)}{(d-1)\ell}\right\} \right]^{-1}$  and  $\ell = \log_4(T)$ . Since  $\mathbf{V} \rightarrow Y \rightarrow \hat{\mathbf{V}}$  forms a Markov chain, Fano's inequality tells us that

$$\begin{aligned} \Pr(\hat{\mathbf{V}} \neq \mathbf{V}) &\geq 1 - \frac{I(\mathbf{V}; Y) + \log 2}{\log |\mathcal{V}'|} \\ &\geq 1 - \frac{1}{(d-1)\ell} \left\{ \frac{(d-1)\ell}{4} + \sum_{j=1}^M \left[ (d-1)\ell(h_2(e^{-a^2}) + \ell e^{-a^2}) \right] + \log 2 \right\} \\ &\geq 1 - \left\{ \sum_{j=1}^M \left[ \frac{1}{4M} \min\left\{1, \frac{H(Y_j)}{(d-1)\ell}\right\} + \frac{6}{80M^2\ell^2} + \frac{1}{256M^4\ell^3} \right] + \frac{\log 2}{(d-1)\ell} \right\}. \end{aligned}$$

In the last step, we used the value of  $a$  along with the fact that  $h_2(p) \leq 1.2\sqrt{p}$  for all  $p \in [0, 1]$ . For  $T \geq 2048$ , we have that  $\Pr(\hat{\mathbf{V}} \neq \mathbf{V}) \geq \frac{4}{5} - \frac{1}{4} = \frac{11}{20} \geq \frac{1}{3}$ . Hence, we can conclude that the estimation error is at least  $\frac{C}{T \sum_{j=1}^M \min\left\{1, \frac{H(Y_j)}{(d-1)\ell}\right\}}$  for some universal constant  $C > 0$  with probability at least  $1/3$ . Consequently, unless  $\sum_{j=1}^M \min\left\{1, \frac{H(Y_j)}{(d-1)\ell}\right\}$  is  $\Omega(M(d-1)\ell)$ , the estimation error will satisfy  $\Omega(1/MT)$ .

with probability at least  $1/3$ . This implies that for at least  $\Omega(M)$  agents  $H(Y_j)$  should be at least as large as  $(d - 1)\ell$  bits. In other words, the uplink cost (per agent) should be at least  $\Omega(d \log T)$  bits since  $\ell = \log_4 T$ .

### D.3 Sparse PLS

We first briefly comment on the choice of the regularization parameter,  $\lambda_k$  and the number of actions at each agent,  $m$  as their choice plays an important in guarantees of Sparse-PLS.  $\lambda_k$  is chosen based on analysis of the LASSO estimator [237] while  $m$  is chosen to ensure that  $X$  satisfies the restricted eigenvalue condition [35]. In particular, based on the result in [25], our choice of  $m = 80(s \log(150d/s) + \log(4/\delta))$  ensures that with probability at least  $1 - \delta/2$  over the randomness of  $\mathcal{B}_s$  the following holds for any  $\theta \in C_s$ ,

$$\frac{3}{4}\|\theta\|_2 \leq \sqrt{\frac{d}{m}}\|X\theta\|_2 \leq \frac{5}{4}\|\theta\|_2.$$

In the above definition  $C_s = \{\theta : \|\theta_{S^c}\|_1 \leq 3\|\theta_S\|_1 \text{ for all } S \subset \{1, 2, \dots, d\} \text{ with } |S| \leq s\}$  where  $\theta_S$  refers to the sub-vector of  $\theta$  corresponding to the coordinates indexed by  $S$ .

Before presenting the proof of Theorem 5.6.1, we state and prove a lemma analogous to Lemma D.1.2 for sparse bandits which will be used later in the proof.

**Lemma D.3.1.** *For any epoch  $k$  in Sparse-PLS, the estimate  $\hat{\theta}_k^{(\text{SERV})}$  satisfies*

$$\|\hat{\theta}_k^{(\text{SERV})} - \theta^*\| \leq \tau_k,$$

*with probability at least  $1 - \delta/2$ .*

*Proof.* Similar to the proof of Lemma D.1.2, we use induction and also define  $\bar{\theta}_{k-1} := 0$  for all epochs  $k$  during the norm estimation stage.

In Sparse-PLS, the estimate  $\hat{\theta}_k^{(\text{SERV})}$  is obtained by minimizing

$$\begin{aligned}
\mathcal{L}_k(\theta) &:= \frac{d}{m} \left\| \frac{1}{M} \sum_{j=1}^M Q(\tilde{\theta}_k^{(j)}) - X(\theta - \bar{\theta}_{k-1}) \right\|_2^2 + \lambda_k \|\theta\|_1 \\
&= \frac{d}{m} \left\| \frac{1}{M} \sum_{j=1}^M (\tilde{\theta}_k^{(j)} + \eta_k^{(j)}) - X(\theta - \bar{\theta}_{k-1}) \right\|_2^2 + \lambda_k \|\theta\|_1 \\
&= \frac{d}{m} \left\| \frac{1}{M} \sum_{j=1}^M (\hat{\theta}_k^{(j)} - X\bar{\theta}_{k-1}) \mathbb{1}_{R_k+B_k} + \frac{1}{M} \sum_{j=1}^M \eta_k^{(j)} - X(\theta - \bar{\theta}_{k-1}) \right\|_2^2 + \lambda_k \|\theta\|_1 \\
&= \frac{d}{m} \left\| \frac{1}{M} \sum_{j=1}^M (\hat{\theta}_k^{(j)} - X\bar{\theta}_{k-1}) \mathbb{1}_{R_k+B_k} + \frac{1}{M} \sum_{j=1}^M \eta_k^{(j)} - X(\theta - \bar{\theta}_{k-1}) \right\|_2^2 + \lambda_k \|\theta\|_1 \\
&= \frac{d}{m} \left\| X\theta^* + \frac{1}{M} \sum_{j=1}^M \Delta_k - X\bar{\theta}_{k-1} + \frac{1}{M} \sum_{j=1}^M \eta_k^{(j)} - X(\theta - \bar{\theta}_{k-1}) \right\|_2^2 + \lambda_k \|\theta\|_1 \\
&= \frac{d}{m} \left\| X\theta^* + \frac{1}{M} \sum_{j=1}^M \Delta_k + \frac{1}{M} \sum_{j=1}^M \eta_k^{(j)} - X\theta \right\|_2^2 + \lambda_k \|\theta\|_1,
\end{aligned}$$

where  $\Delta_k$  corresponds to clipped sub-Gaussian observation noise. Using the result of Theorem 2.18 in Rigollet [237] and our choice of  $m$ , we can conclude that

$$\|\hat{\theta}_k^{(\text{SERV})} - \theta^*\|_2^2 \leq \frac{1024sd \log(8K/\delta)}{mM} \left( \frac{\sigma^2}{s_k} + \frac{\alpha_k^2}{4s} \right).$$

Plugging in the choice of  $s_k$  and  $\alpha_k$  yields,

$$\|\hat{\theta}_k^{(\text{SERV})} - \theta^*\|_2 \leq \frac{3}{\sqrt{M}} \cdot 2^{-(k+1)} = \tau_k,$$

with probability at least  $1-\delta/K$ . Here, similar to proof of Lemma D.1.2 the choice of  $R_k$  and  $B_k$  allows us to use the clipped sub-Gaussian concentration which is implicitly used while invoking the result from Rigollet [237].

Using an argument similar to the one in Lemma D.1.2, we can conclude that

the above inequality holds for epochs during the algorithm with probability at least  $1 - \delta/2$ .

□

### D.3.1 Proof of Theorem 5.6.1

The analysis for the regret performance of Sparse-PLS is almost identical to that of PLS, as described in Appendix D.1. Once again, the regret is decomposed into the sum of regret incurred during the norm estimation stage and the refinement stage.

Recall that the regret during the norm estimation stage of PLS is bounded using the result in Lemma D.1.2. Since Lemma D.3.1 is identical to Lemma D.1.2, the proof of Lemma 5.4.2 follows through almost unchanged for the case of Sparse-PLS. The only difference in the case of Sparse-PLS is that there are  $m$  actions in the basis set instead of  $d$  as in the case of PLS. This scales the corresponding term in the regret incurred during the norm estimation stage by a factor of  $m/d$  resulting in an overall regret of  $O(\sqrt{sdMT} \log(d/\delta) \log(MT) \log(\log T/\delta))$ .

Similarly, the analysis in the refinement stage also follows through identically but for a couple of minor changes. The regret during a exploration sub-epoch is also scaled by a factor of  $m/d$  for the same reason as in the case of the norm estimation stage. The analysis for exploration sub-epoch follows through as is with the different value of  $t_k$  which is also scaled by a factor of  $s/d$ . Carrying out the same steps with these updated values result in an overall regret of  $O(\sqrt{sdMT} \log(d/\delta) \log(MT) \log(\log T/\delta))$ , as required.

For the communication cost, the bound on the downlink cost follows as in case for PLS. For the case of the uplink cost, note that in Sparse-PLS, the transmitted vector lies in  $\mathbb{R}^m$ . Using the same argument in used on the proof of Lemma 5.4.5 for vectors in  $\mathbb{R}^m$  along with the updated choice of  $R_k$ ,  $B_k$  and  $\alpha_k$ , we can conclude that each uplink message is at most  $O(m)$  bits long, resulting in an overall uplink cost of  $O(m \log T) = O(s \log dT)$ .

## D.4 CEAL

We begin with setting up some notation that will be used later in the proofs.

Consider the task of estimating the norm of a vector  $y$  with  $\|y\| \leq 1$  using the Norm Estimation Routine described in Algorithm 6, including the quantization step carried out using the quantization process described in Section 5.3.2. Let  $\eta_j^{(m)} \in \mathbb{R}^d$  denote the quantization noise added by  $m^{\text{th}}$  client during the  $j^{\text{th}}$  epoch, i.e., the quantized version received by the server can be written as  $Q(\hat{y}_j^{(m)}) = \hat{y}_j^{(m)} + \eta_j^{(j)}$ .

Let  $\mathcal{E}_j$  denote the event where the observations satisfy the following two inequalities:

$$\begin{aligned} \left\| \frac{1}{M} \sum_{m=1}^M \eta_j^{(m)} \right\| &\leq \frac{2\gamma_j}{\sqrt{M}} \left( 1 + \sqrt{\frac{1}{2d} \log \left( \frac{4j^2}{\delta} \right)} \right) \\ \left\| \frac{1}{M} \sum_{m=1}^M \hat{y}_j^{(m)} - y \right\| &\leq \frac{4\sigma}{\sqrt{Ms_j}} \left( 1 + \sqrt{\frac{1}{2d} \log \left( \frac{4j^2}{\delta} \right)} \right). \end{aligned}$$

Similarly, define

$$\mathcal{E} = \cap_{j \geq 1} \mathcal{E}_j. \quad (\text{D.2})$$

This definition is similar to the event defined in Eqn. (D.1). We have redefined it for convenience and to allow for the minor differences in conventions between distributed linear bandits and distributed stochastic bandits.

Using the results in Appendix D.1, we can conclude that  $\mathcal{E}_j$  holds with probability at least  $1 - \delta/(2j^2)$ . Consequently, an application of the union bound yields  $\Pr(\mathcal{E}) \geq 1 - \delta \sum_{j=1}^{\infty} (2j^2)^{-1} \geq 1 - \delta$ .

Similarly, using the result proved in Appendix D.1, we can conclude that conditioned on the event  $\mathcal{E}$ ,

$$\|\hat{y}_j^{(\text{SERV})} - y\| \leq 2^{-j} + 2^{-(j+1)} = 3 \cdot 2^{-(j+1)} = \tau_j,$$

holds for all  $j \geq 1$ .

We now move on to the proofs of Lemmas 5.7.2 and 5.7.3.

#### D.4.1 Proof of Lemma 5.7.2

Consider the  $k^{\text{th}}$  iterate,  $x^{(k)}$ . Let  $g(x^{(k)}) := \nabla f(x^{(k)})$  denote the true gradient at  $x^{(k)}$ . Recall that the  $\hat{g}_j^{(\text{SERV})}(x^{(k)})$  denotes the estimate of the gradient at the server at the end of the  $j^{\text{th}}$  epoch. For brevity of notation, we drop the argument  $x^{(k)}$  throughout this proof.

Under the event  $\mathcal{E}$  as defined in Eqn. (D.2), the Norm Estimation Routine at this query point will terminate at the end of epoch  $j_0$ , where  $j_0 := \min\{j \in \mathbb{N} : 4\tau_j \leq \|\hat{g}_j^{(\text{SERV})}\|\}$ . We note that for all  $j$  for which the inequality  $4\tau_j \leq \|\hat{g}_j^{(\text{SERV})}\|$

holds, we also have the relation

$$\begin{aligned}\|\hat{g}_j^{(\text{SERV})} - g\| &\leq \tau_j \leq \frac{1}{4} \cdot \|\hat{g}_j^{(\text{SERV})}\| \\ \implies \|g\| &\in \left[ \frac{3}{4} \|\hat{g}_j^{(\text{SERV})}\|, \frac{5}{4} \|\hat{g}_j^{(\text{SERV})}\| \right] \\ \implies \|\hat{g}_j^{(\text{SERV})}\| &\in \left[ \frac{4}{5} \|g\|, \frac{4}{3} \|g\| \right].\end{aligned}$$

Consequently, we also have  $\tau_j \leq \frac{1}{3} \cdot \|g\|$  which implies that  $j \geq \log_2(9/2\|g\|)$ . Since  $j_0$  is the smallest natural number satisfying this relation,  $j_0 = \lceil \log_2(9/2\|g\|) \rceil$ .

Thus, under the event  $\mathcal{E}$ , the Norm Estimation Routine terminates at the end of epoch  $j_0$ . Consequently, we can bound  $t_k$  as

$$\begin{aligned}t_k &\leq \sum_{j=1}^{j_0} s_j \leq \sum_{j=1}^{j_0} \left[ \frac{40\sigma^2}{M} \cdot \log(16Mj^2/\delta) \cdot 4^j + 1 \right] \\ &\leq \frac{160\sigma^2}{3M} \cdot \log(16Mj_0^2/\delta) \cdot 4^{j_0} + j_0.\end{aligned}$$

On plugging in the value of  $j_0$ , we obtain that  $t_k$  satisfies  $O\left(\frac{1}{M\|\nabla f(x^{(k)})\|^2}\right)$ .

Similarly, we also can obtain a lower bound on  $t_k$ . Note that  $t_k \geq s_{j_0}$ . Substituting the expression for  $s_j$  and the value of  $j_0$  yields us that  $t_k$  satisfies  $\Omega\left(\frac{1}{M\|\nabla f(x^{(k)})\|^2}\right)$ .

#### D.4.2 Proof of Lemma 5.7.3

To establish an upper bound on the number of epochs (and communication rounds), we first establish a convergence rate of the iterates. In particular, we show that conditioned on the event  $\mathcal{E}$ , the magnitude of the gradient across iterates decreases exponentially fast, resulting in a logarithmic number of communication rounds. We use the same notation as described in Appendix D.4.1.

In addition, we use  $[\hat{g}]_j^{(\text{SERV})}(x^{(k)})$  to denote the quantized version of  $\hat{g}_j^{(\text{SERV})}(x^{(k)})$ , i.e.,  $Q(\hat{g}_j^{(\text{SERV})}(x^{(k)}), \phi_j, B_j + \tau_j)$ , which is sent to the clients. Once again, for brevity of notation, we drop the argument  $x^{(k)}$  from the gradient terms throughout this proof.

The quantization strategy used in CEAL guarantees that  $\|\hat{g}_j^{(\text{SERV})} - [\hat{g}]_j^{(\text{SERV})}\| \leq \tau_j$ . Consequently,  $\|[\hat{g}]_j^{(\text{SERV})} - g\| \leq 2\tau_j \leq \frac{2}{3} \cdot \|g\|$  which leads to the bounds  $\langle g, [\hat{g}]_j^{(\text{SERV})} \rangle \geq \|g\|^2/3$  and  $\|[\hat{g}]_j^{(\text{SERV})}\| \leq 5\|g\|/3$ . Using the  $\beta$ -smoothness of  $f$ , we have,

$$\begin{aligned} f(x^{(k+1)}) &= f(x^{(k)} - \eta[\hat{g}]_j^{(\text{SERV})}(x^{(k)})) \\ &\leq f(x^{(k)}) - \eta \langle g, [\hat{g}]_j^{(\text{SERV})} \rangle + \frac{\eta^2 \beta}{2} \|[\hat{g}]_j^{(\text{SERV})}\|^2 \\ &\leq f(x^{(k)}) - \frac{\eta}{3} \|\nabla f(x^{(k)})\|^2 + \frac{25\eta^2 \beta}{18} \|\nabla f(x^{(k)})\|^2 \\ &\leq f(x^{(k)}) - \frac{\eta}{3} \|\nabla f(x^{(k)})\|^2 + \frac{5\eta}{18} \|\nabla f(x^{(k)})\|^2 \\ &\leq f(x^{(k)}) - \frac{\eta}{18} \|\nabla f(x^{(k)})\|^2, \end{aligned}$$

where the fourth step uses the relation  $\eta \leq 1/(5\beta)$ . Using the  $\alpha$ -strong convexity of  $f$  we obtain,

$$f(x^{(k+1)}) - f(x^*) \leq \left(1 - \frac{\alpha\eta}{9}\right) (f(x^{(k)}) - f(x^*)).$$

Consequently, the sub-optimality gap after  $k$  iterates is given by

$$f(x^{(k)}) - f(x^*) \leq (1 - \alpha\eta/9)^{k-1} (f(x^{(1)}) - f(x^*)).$$

Using  $\beta$ -smoothness of  $f$ , we can translate this bound to a bound on the magnitude of the gradient, i.e.,  $\|\nabla f(x^{(k)})\|^2 \leq C(1 - \alpha\eta/9)^{k-1}$  for some constant  $C > 0$ .

To obtain the bound on the number of epochs, note that the lower bound on  $t_k$  obtained in Appendix D.4.1 suggests that if the inequality  $\|\nabla f(x^{(K)})\|^2 \leq C'/MT$

holds for some iterate  $x^{(K)}$  and a constant  $C' > 0$ , independent of  $M$  and  $T$ , then  $t_K > T$ , implying the algorithm terminates by epoch  $K$ . Using the exponential convergence of the gradient magnitude across iterates, we can conclude that such an epoch index  $K$  satisfies  $O(\log(MT))$ .

APPENDIX E  
CHAPTER 6

### E.1 Proof of Theorem 6.3.1

To bound the regret of CEPE, we consider the regret incurred during the exploration and exploitation epochs separately. The regret corresponding to the exploration epoch is bounded with a constant factor of the cardinality of the exploration epoch, yielding the  $KN_T$  term. The bound on the regret corresponding to the exploitation epoch is obtained in two steps. In the first step, we use the choice of  $x_{i,t}$  and Lemma 6.2.4 to bound  $|f_i(x_i^*) - f(x_{i,t})|$  in terms of the posterior standard deviation at  $x_i^*$  and  $x_{i,t}$ . It is notable that, in this analysis, since  $x_{i,t}$  in the exploration epoch of CEPE are selected in a purely exploratory way, we can use Lemma 6.2.4 which provides tighter confidence intervals compared to the ones used in the analysis of GP-UCB and GP-TS [68]. That is the key in proving an always sublinear regret bound for CEPE. See also Vakili *et al.* [299] for further discussions on the confidence intervals. In the second step, we show that the point selection rule in the exploration epoch, based on maximally reducing the uncertainty, allows us to bound both of the posterior standard deviation terms mentioned above by  $\sqrt{\gamma_{N_t}/N_t}$ , up to an absolute constant factor. We arrive at the theorem by summing the instantaneous regret over  $i$  and  $t$ .

Before bounding the regret, we first obtain an upper bound on the RKHS norm of the objective functions of the clients. For any client  $i \in \{1, 2, \dots, K\}$ , we

have,

$$\begin{aligned}
\|f_i\|_{H_k} &= \left\| \alpha_i h_i + \frac{1 - \alpha_i}{K} \sum_{j=1}^K h_j \right\|_{H_k} \\
&\leq \alpha_i \|h_i\|_{H_k} + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \|h_j\|_{H_k} \\
&\leq \alpha_i B_i + \frac{1 - \alpha_i}{K} \sum_{j=1}^K B_j.
\end{aligned}$$

For each  $i \in \{1, 2, \dots, K\}$  define  $\bar{B}_i := \alpha_i B_i + \frac{1 - \alpha_i}{K} \sum_{j=1}^K B_j$ . Thus,  $\bar{B}_i$  is an upper bound on the RKHS norm of the objective functions.

We first consider the regret incurred during the exploration epoch. Let  $R_1$  denote the regret incurred during the exploration epoch. We then have

$$\begin{aligned}
R_1 &= \sum_{i=1}^K \sum_{t \in \mathcal{A}(T)} f_i(x_i^*) - f_i(x_{i,t}) \\
&\leq \sum_{i=1}^K \sum_{t \in \mathcal{A}(T)} 2\bar{B}_i \\
&\leq \sum_{i=1}^K 2\bar{B}_i |\mathcal{A}(T)| \\
&\leq 2(\max_i \bar{B}_i) K N_T.
\end{aligned}$$

In the second step, we have used  $\sup_{x \in \mathcal{X}} f(x) \leq \|f\|_{H_k}$ , which can be shown as follows

$$\begin{aligned}
\sup_{x \in \mathcal{X}} f(x) &= \sup_{x \in \mathcal{X}} \langle f, k(\cdot, x) \rangle \\
&\leq \sup_{x \in \mathcal{X}} \|f\|_{H_k} \|k(\cdot, x)\|_{H_k} \\
&\leq \left( \sup_{x \in \mathcal{X}} k(x, x) \right) \|f\|_{H_k}.
\end{aligned}$$

Using  $k(x, x) \leq 1$  for all  $x \in \mathcal{X}$  gives us the required bound.

We now consider the regret incurred during the exploitation epoch, which we denote by  $R_2$ . We first consider the regret incurred by a single client  $i$ . Based on Assumption 6.2.3, we consider a discretization  $\mathcal{D}_T$  with  $|\mathcal{D}_T| = O(T^d)$  such that for all  $x \in \mathcal{X}$ ,  $i \in \{1, 2, \dots, K\}$  and  $t \leq T$ , we have,  $|h_i(x) - h_i([x]_{\mathcal{D}_T})| \leq N_T^{-1/2}$  and  $|\mu_t^{(h_i)}(x) - \mu_t^{(h_i)}([x]_{\mathcal{D}_T})| \leq N_T^{-1/2}$ , where  $[x]_{\mathcal{D}_T}$  denotes the point closest to  $x$  on the discretization  $\mathcal{D}_T$ . Note that we do not need to explicitly construct such a discretization for the actual algorithm. It is considered only for the purpose of analysis.

Using Lemma 6.2.4 and a union bound over all the points in the discretization  $\mathcal{D}_T$ , we can conclude that with probability at least  $1 - \delta_0/K$ ,  $|h_i(x') - \mu_{t-1}^{(h_i)}(x')| \leq \beta(B_i, \delta'_0) \sigma_{t-1}^{(h_i)}(x')$  holds for all  $x' \in \mathcal{D}_T$ ,  $t \leq T$  and given client  $i \in \{1, 2, \dots, K\}$  where  $\delta'_0 = \delta_0/(2KT|\mathcal{D}_T|)$ . Consequently, for any client  $i$  we have,

$$\begin{aligned} |f_i(x') - \mu_{t-1}^{(f_i)}(x')| &= \left| \left( \alpha_i h_i(x') + \frac{1 - \alpha_i}{K} \sum_{j=1}^K h_j(x') \right) - \left( \alpha_i \mu_{t-1}^{(h_i)}(x') + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \mu_{t-1}^{(h_j)}(x') \right) \right| \\ &\leq \alpha_i |h_i(x') - \mu_{t-1}^{(h_i)}(x')| + \frac{1 - \alpha_i}{K} \sum_{j=1}^K |h_j(x') - \mu_{t-1}^{(h_j)}(x')| \\ &\leq \alpha_i \cdot \beta(B_i, \delta'_0) \cdot \sigma_{t-1}^{(h_i)}(x') + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \beta(B_j, \delta'_0) \cdot \sigma_{t-1}^{(h_j)}(x'). \end{aligned}$$

The above relation holds for all  $i \in \{1, 2, \dots, K\}$ ,  $x' \in \mathcal{D}_T$ ,  $t \leq T$  with probability at least  $1 - \delta_0$ , which follows immediately from the union bound. With a slight abuse of notation, we define  $\sigma_{t-1}^{(f_i)}(x') := \alpha_i \cdot \beta(B_i, \delta'_0) \cdot \sigma_{t-1}^{(h_i)}(x') + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \beta(B_j, \delta'_0) \cdot \sigma_{t-1}^{(h_j)}(x')$ .

With this relation, we move onto bound the regret incurred by any client  $i$

during the exploitation epoch. We have,

$$\begin{aligned}
\sum_{t \notin \mathcal{A}(T)} f_i(x_i^*) - f_i(x_{i,t}) &= \sum_{t \notin \mathcal{A}(T)} f_i(x_i^*) - f_i([x_{i,t}]_{\mathcal{D}_T}) + f_i([x_{i,t}]_{\mathcal{D}_T}) - f_i(x_{i,t}) \\
&\leq \sum_{t \notin \mathcal{A}(T)} f_i(x_i^*) - f_i([x_{i,t}]_{\mathcal{D}_T}) + N_T^{-1/2} \\
&\leq \sum_{t \notin \mathcal{A}(T)} f_i(x_i^*) - \mu_{t-1}^{(f_i)}(x_i^*) + \mu_{t-1}^{(f_i)}(x_{i,t}) - f_i([x_{i,t}]_{\mathcal{D}_T}) + N_T^{-1/2} \\
&\leq \sum_{t \notin \mathcal{A}(T)} \left[ f_i(x_i^*) - \mu_{t-1}^{(f_i)}(x_i^*) + \mu_{t-1}^{(f_i)}(x_{i,t}) - \mu_{t-1}^{(f_i)}([x_{i,t}]_{\mathcal{D}_T}) \right. \\
&\quad \left. + \mu_{t-1}^{(f_i)}([x_{i,t}]_{\mathcal{D}_T}) - f_i([x_{i,t}]_{\mathcal{D}_T}) + N_T^{-1/2} \right] \\
&\leq \sum_{t \notin \mathcal{A}(T)} f_i(x_i^*) - \mu_{t-1}^{(f_i)}(x_i^*) + \mu_{t-1}^{(f_i)}([x_{i,t}]_{\mathcal{D}_T}) - f_i([x_{i,t}]_{\mathcal{D}_T}) + 2N_T^{-1/2} \\
&\leq \sum_{t \notin \mathcal{A}(T)} \sigma_{t-1}^{(f_i)}(x_i^*) + \sigma_{t-1}^{(f_i)}([x_{i,t}]_{\mathcal{D}_T}) + 2N_T^{-1/2}.
\end{aligned}$$

In the third step, we used the fact that  $x_{i,t} = \arg \max_{x \in \mathcal{X}} \mu_{t-1}^{(f_i)}(x)$ . The expression on the RHS in the above equation is bounded using the following lemma.

**Lemma E.1.1.** *Consider the epoch of points  $\{x_1, x_2, \dots, x_t\}$  generated using the maximum posterior variance sampling scheme, that is,  $x_t = \arg \max_{x \in \mathcal{X}} \sigma_{t-1}(x)$ , where  $\sigma_{t-1}^2$  is the posterior variance computed from the history  $\{x_1, x_2, \dots, x_{t-1}\}$ . Then, at any time  $t$ , the following inequality is true.*

$$\sup_{x \in \mathcal{X}} \sigma_t(x) \leq \sqrt{\frac{12\gamma_t}{t}},$$

where  $\gamma_t$  is the information gain after  $t$  steps.

The lemma provides an upper bound on the posterior standard deviation under the maximum posterior variance sampling scheme, as adopted in the exploration epoch of CEPE. The proof of this lemma is provided at the end of this section.

Note that  $t \notin \mathcal{A}(T) \implies t \notin \mathcal{A}(t)$  which in turn implies that  $|\mathcal{A}(t)| > N_t$ . Thus, at any time instant  $t$  in the exploitation epoch, the posterior standard deviation is computed based on a history of at least  $N_t$  observations taken according to the maximum posterior variance sampling scheme. Hence, using Lemma E.1.1 we can conclude that  $\sigma_{t-1}^{(h_i)}(x) \leq \sqrt{12\gamma_{N_t}/N_t}$  for all  $x \in \mathcal{X}$  and all clients  $i \in \{1, 2, \dots, K\}$ . Consequently, for any client  $i$  and any  $x \in \mathcal{X}$ ,

$$\begin{aligned}\sigma_{t-1}^{(f_i)}(x) &= \alpha_i \cdot \beta(B_i, \delta'_0) \cdot \sigma_{t-1}^{(h_i)}(x) + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \beta(B_j, \delta'_0) \cdot \sigma_{t-1}^{(h_j)}(x) \\ &\leq \alpha_i \cdot \beta(B_i, \delta'_0) \cdot \sqrt{\frac{12\gamma_{N_t}}{N_t}} + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \beta(B_j, \delta'_0) \cdot \sqrt{\frac{12\gamma_{N_t}}{N_t}} \\ &\leq \sqrt{\frac{12\gamma_{N_t}}{N_t}} \cdot \left( \alpha_i \cdot \beta(B_i, \delta'_0) + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \beta(B_j, \delta'_0) \right) \\ &\leq \sqrt{\frac{12\gamma_{N_t}}{N_t}} \cdot \left( \alpha_i \cdot \beta(B_i, \delta'_0) + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \beta(B_j, \delta'_0) \right) \\ &= \sqrt{\frac{12\gamma_{N_t}}{N_t}} \cdot \beta(\bar{B}_i, \delta'_0),\end{aligned}$$

where the last step follows from the definition of  $\beta(B, \delta)$  and  $\bar{B}_i$ . On plugging this bound in the expression for the regret incurred by a single client during the exploitation epoch, we obtain the following bound on  $R_2$ , the overall regret incurred during the exploitation epoch, which holds with probability of at least  $1 - \delta_0$ .

$$\begin{aligned}R_2 &= \sum_{i=1}^K \sum_{t \notin \mathcal{A}(T)} f_i(x_i^*) - f_i(x_{i,t}) \\ &\leq \sum_{i=1}^K \sum_{t \notin \mathcal{A}(T)} \sigma_{t-1}^{(f_i)}(x_i^*) + \sigma_{t-1}^{(f_i)}([x_{i,t}]_{\mathcal{D}_T}) + 2N_T^{-1/2} \\ &\leq \sum_{i=1}^K \sum_{t \notin \mathcal{A}(T)} \left( \sqrt{\frac{12\gamma_{N_t}}{N_t}} \cdot \beta(\bar{B}_i, \delta'_0) + \sqrt{\frac{12\gamma_{N_t}}{N_t}} \cdot \beta(\bar{B}_i, \delta'_0) + 2N_T^{-1/2} \right) \\ &\leq 2KT \left[ \left( \max_i \bar{B}_i + R \sqrt{\frac{2}{\lambda} \log\left(\frac{2}{\delta'_0}\right)} \right) \sqrt{\frac{12\gamma_{N_T}}{N_T}} + \frac{1}{\sqrt{N_T}} \right],\end{aligned}$$

The last step follows by noting that

$$\sum_{t \notin \mathcal{A}(T)} \sqrt{\frac{\gamma_{N_t}}{N_t}} \leq T \cdot \left( \frac{1}{T} \sum_{t=1}^T \sqrt{\frac{\gamma_{N_t}}{N_t}} \right) \leq T \cdot \sqrt{\frac{\gamma_{N_T}}{N_T}},$$

where the last inequality is a result of applying Jensen's inequality on the concave function  $t \rightarrow \sqrt{\gamma_t/t}$  and the monotonicity of  $N_t$ .

Adding the bounds on  $R_1$  and  $R_2$ , we arrive at the theorem.

### E.1.1 Proof of Lemma E.1.1

The proof is based on Lemma 4 of Chowdhury and Gopalan [68] which bounds the sum of sequential posterior standard deviations of a GP model as follows

$$\sum_{s=1}^t \sigma_{s-1}(x_s) \leq \sqrt{12t\gamma_t}.$$

This is true for all  $t \geq 1$  and for any sampling scheme that sequentially generates the points  $\{x_1, x_2, \dots\}$ . In the case of maximally reducing the variance, for any choices of  $s' \leq s$ , we have

$$\begin{aligned} \sigma_{s'-1}(x_{s'}) &\geq \sigma_{s'-1}(x_s) \\ &\geq \sigma_{s-1}(x_s). \end{aligned}$$

The first inequality comes from the selection rule. The second inequality comes from the fact that conditioning a GP on a larger set of points reduces the variance (that follows from the positive definiteness of the covariance matrix).

This implies that  $\sigma_{t-1}(x_t)$  is the smallest term in the sum  $\sum_{s=1}^t \sigma_{s-1}(x_s)$  and hence has to be smaller than the mean. Consequently, we have,

$$\sup_{x \in \mathcal{X}} \sigma_{t-1}(x) = \sigma_{t-1}(x_t) \leq \frac{1}{t} \sum_{s=1}^t \sigma_{s-1}(x_s) \leq \sqrt{\frac{12\gamma_t}{t}},$$

as required.

## E.2 Proof of Theorem 6.4.2

We first introduce some additional notation corresponding the feature representation of functions in a RKHS, which is used in a part of the proof. In particular, the kernel function  $k(\cdot, \cdot)$  is associated with a nonlinear feature map  $\phi(\cdot) : \mathbb{R}^d \rightarrow \mathcal{H}_k$  such that  $k(x, x') = \phi(x')^\top \phi(x)$ . Consequently, the reproducing property can also be written as  $h(x) = \phi(x)^\top h$ . Corresponding to a client  $i$ , at each time instant  $t$ , we define the matrix  $\Phi_{i,t} = [\phi(x_{i,1}), \phi(x_{i,2}), \dots, \phi(x_{i,t})]^\top$  based on the points sampled by client  $i$ , i.e.,  $\mathbf{x}_{i,t}$ .

Let  $\mathbf{z}_{i,t} = \{z_{i,1}, z_{i,2}, \dots, z_{i,m_i}\}$  denote the set of inducing points chosen from  $\mathbf{x}_{i,t}$  according to the strategy outlined in Section 6.4. Let  $S_{i,t}$  be the diagonal  $\mathbb{R}^{t \times t}$  matrix with  $W_{i,j}/\sqrt{p_{i,j}}$  as the  $j^{\text{th}}$  element on the diagonal. Recall that  $p_{i,j} = q_0[\sigma_{t-1}^{(h_i)}(x_{i,j})]^2$  was the probability of choosing point  $x_{i,j}$  for  $j = 1, 2, \dots, t$  and  $W_{i,j}$  was the corresponding  $\{0, 1\}$  random variable that determined whether that point was a part of the inducing set  $\mathbf{z}_{i,t}$  or not. Consequently, we have the following relation -

$$\sum_{z \in \mathbf{z}_{i,t}} \frac{1}{p_{i,j}} \phi(z) \phi(z)^\top = \sum_{j=1}^t \frac{W_{i,j}}{p_{i,j}} \phi(x_{i,j}) \phi(x_{i,j})^\top = \Phi_{i,t} S_{i,t} S_{i,t}^\top \Phi_{i,t}^\top.$$

Furthermore, if  $\tilde{H}_k$  denotes the subspace of  $H_k$  spanned by the inducing points, then using  $S_{i,t}$ , we can also define the projection operator onto this subspace, which we denote by  $P_{i,t}$ . In particular, we have,

$$P_{i,t} = \Phi_{i,t} S_{i,t} (S_{i,t}^\top \Phi_{i,t}^\top \Phi_{i,t} S_{i,t})^+ S_{i,t}^\top \Phi_{i,t}^\top,$$

where  $A^+$  denotes the pseudo-inverse of the matrix  $A$ . Using  $P_{i,t}$ , we define an approximate feature map  $\tilde{\phi}(\cdot) : P_{i,t} \phi(\cdot)$  that corresponds to the subspace  $\tilde{H}_k$ . Similar to  $\Phi_{i,t}$ , we define  $\tilde{\Phi}_{i,t} = [\tilde{\phi}(x_{i,1}), \tilde{\phi}(x_{i,2}), \dots, \tilde{\phi}(x_{i,t})]^\top = \Phi_{i,t} P_{i,t}$ . Lastly, we also define two more matrices,  $A_{i,t} = \Phi_{i,t}^\top \Phi_{i,t} + \lambda I$  and  $\tilde{A}_{i,t} = \tilde{\Phi}_{i,t}^\top \tilde{\Phi}_{i,t} + \lambda I$ .

With the notations set up, we move on to proving Theorem 6.4.2. The basic proof follows the same structure as that of Theorem 6.3.1. We consider the regret separately in the exploration, communication and the exploitation sequence. For the exploration and the communication sequence, the regret is bounded with a constant factor of their lengths. For each client  $i$ , since the algorithm queries the same point,  $\tilde{x}_i^* = \arg \max_{x \in \mathcal{X}} \tilde{\mu}_{N_T}^{(f_i)}(x)$  throughout the exploitation sequence, an upper bound on the regret incurred during the exploitation sequence is simply obtained by bounding the error  $f_i(x^*) - f_i(\tilde{x}_{i,N_T}^*)$  and multiplying with  $T$ , the length of the time horizon. The bound on  $f_i(x_i^*) - f_i(\tilde{x}_{i,N_T}^*)$  is obtained using Lemma 6.4.3 using argument similar to the ones used in the proof of Theorem 6.3.1. The overall regret is then obtained by summing the contributions of these two different terms.

We first consider the regret incurred during the exploration and communication phases. Let  $R_1$  denote the regret incurred during this period. We then have

$$\begin{aligned} R_1 &= \sum_{i=1}^K \sum_{t=1}^{N_T+N_T^{(c)}} f_i(x_i^*) - f_i(x_{i,t}) \\ &\leq \sum_{i=1}^K \sum_{t=1}^{N_T+N_T^{(c)}} 2\bar{B}_i \\ &\leq \sum_{i=1}^K 2\bar{B}_i(N_T + N_T^{(c)}) \\ &\leq 2 \cdot (\max_i \bar{B}_i) \cdot K(N_T + 9(1 + 1/\lambda)q_0\gamma_{N_T}) \\ &\leq 2 \cdot (1 + 9(1 + 1/\lambda)q_0) \cdot (\max_i \bar{B}_i) \cdot KN_T. \end{aligned}$$

In the second step, we have once again used  $\sup_{x \in \mathcal{X}} h(x) \leq \|h\|_{H_k}$ .

To bound the regret incurred during the exploitation sequence, denoted by  $R_2$ , we consider a discretization  $\mathcal{D}_{N_T}$  with  $\mathcal{D}_{N_T} = O(T^d)$  of the domain

$\mathcal{X}$ . Based on Assumption 6.2.3, we have that for all  $x \in \mathcal{X}$ ,  $i \in \{1, 2, \dots, K\}$ ,  $|h_i(x) - h_i([x]_{\mathcal{D}_{N_T}})| \leq N_T^{-1/2}$  and  $|\tilde{\mu}_{N_T}^{(h_i)}(x) - \tilde{\mu}_{N_T}^{(h_i)}([x]_{\mathcal{D}_{N_T}})| \leq N_T^{-1/2}$ , where  $[x]_{\mathcal{D}_{N_T}}$  denotes the point closest to  $x$  on the discretization  $\mathcal{D}_{N_T}$ . As before, we do not need to explicitly construct such a discretization for the actual algorithm.

Using Lemma 6.4.3, a union bound over all the points in the discretization  $\mathcal{D}_{N_T}$  and a series of argument similar to the ones used in the proof of Theorem 6.3.1 (See Appendix E.1), we can conclude that the following relations hold for all clients  $i \in \{1, 2, \dots, K\}$  for any given fixed  $x' \in \mathcal{D}_{N_T}$  with probability at least  $1 - \delta_0/3$ :

$$|f_i(x') - \tilde{\mu}_{N_T}^{(f_i)}(x')| \leq \alpha_i \cdot \tilde{\beta}(B_i, \delta_0'') \cdot \tilde{\sigma}_{N_T}^{(h_i)}(x') + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \tilde{\beta}(B_j, \delta_0'') \cdot \tilde{\sigma}_{N_T}^{(h_j)}(x')$$

and,

$$f_i(x_i^*) - f_i(\tilde{x}_{i,N_T}^*) \leq \tilde{\sigma}_{N_T}^{(f_i)}(x_i^*) + \tilde{\sigma}_{N_T}^{(f_i)}([\tilde{x}_{N_T}^*]_{\mathcal{D}_{N_T}}) + 2N_T^{-1/2},$$

where  $\delta_0'' = \delta_0/(6K|\mathcal{D}_{N_T}|)$ ,  $\tilde{\sigma}_{N_T}^{(f_i)}(x) := \alpha_i \cdot \tilde{\beta}(B_i, \delta_0'') \cdot \tilde{\sigma}_{N_T}^{(h_i)}(x) + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \tilde{\beta}(B_j, \delta_0'') \cdot \tilde{\sigma}_{N_T}^{(h_j)}(x)$

and  $\tilde{x}_{i,N_T}^* = \arg \max_{x \in \mathcal{X}} \tilde{\mu}_{N_T}^{(f_i)}(x)$ .

To bound  $\tilde{\sigma}_{N_T}^{(f_i)}(\cdot)$ , we first bound our *approximate* predictive standard deviation  $\tilde{\sigma}_{N_T}^{(h_j)}(\cdot)$  for all  $j \in \{1, 2, \dots, K\}$  based on our construction of the set of inducing points. For the particular choice of  $q_0$  as used in Lemma 6.4.1, the authors in [49] establish the following relation which holds with probability at least  $1 - \delta_0/3$  for all clients  $j \in \{1, 2, \dots, K\}$ :

$$\frac{1}{\chi} [\sigma_{N_T}^{(h_j)}(x)]^2 \leq [\tilde{\sigma}_{N_T}^{(h_j)}(x)]^2 \leq \chi [\sigma_{N_T}^{(h_j)}(x)]^2,$$

where  $\sigma_{N_T}^2(x)$  is the true predictive variance based on all the  $N_T$  points queried during the exploration phase and  $\chi = (1 + \varepsilon)/(1 - \varepsilon)$ . This implies that *approximate*

posterior variance and the *actual* posterior variance are within a constant factor of each other. On using this relation along with Lemma E.1.1, we can conclude

$$\begin{aligned}
\tilde{\sigma}_{N_T}^{(f_i)}(x) &= \alpha_i \cdot \tilde{\beta}(B_i, \delta_0'') \cdot \tilde{\sigma}_{N_T}^{(h_i)}(x) + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \tilde{\beta}(B_j, \delta_0'') \cdot \tilde{\sigma}_{N_T}^{(h_j)}(x) \\
&\leq \alpha_i \cdot \tilde{\beta}(B_i, \delta_0'') \cdot \chi \cdot \sigma_{N_T}^{(h_i)}(x) + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \tilde{\beta}(B_j, \delta_0'') \cdot \chi \cdot \sigma_{N_T}^{(h_j)}(x) \\
&\leq \chi \sqrt{\frac{12\gamma_{N_T}}{N_T}} \cdot \left( \alpha_i \cdot \tilde{\beta}(B_i, \delta_0'') + \frac{1 - \alpha_i}{K} \sum_{j=1}^K \tilde{\beta}(B_j, \delta_0'') \right) \\
&\leq \chi \sqrt{\frac{12\gamma_{N_T}}{N_T}} \cdot \tilde{\beta}(\bar{B}_i, \delta_0'').
\end{aligned}$$

Consequently,  $R_2$  can be bounded as

$$\begin{aligned}
R_2 &= \sum_{i=1}^K \sum_{t=N_T+N_T^c+1}^T f_i(x_i^*) - f_i(\tilde{x}_{i,N_T}^*) \\
&\leq T \sum_{i=1}^K \left( \tilde{\sigma}_{N_T}^{(f_i)}(x_i^*) + \tilde{\sigma}_{N_T}^{(f_i)}([\tilde{x}_{N_T}^*]_{\mathcal{D}_{N_T}}) + 2N_T^{-1/2} \right) \\
&\leq 2KT \left[ \tilde{\beta}(\bar{B}_i, \delta_0'') \sqrt{\frac{12\gamma_{N_T}}{N_T}} + \frac{1}{\sqrt{N_T}} \right].
\end{aligned}$$

Adding the bound on  $R_1$  and  $R_2$  yields the required result.

### E.2.1 Proof of Lemma 6.4.3

Throughout the proof, we use the same notation as established earlier with the subscript corresponding to the client dropped for simplicity. We begin considering the definition of  $\tilde{\mu}_t$ . We have,

$$\begin{aligned}
h(x) - \tilde{\mu}_t(x) &= h(x) - k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} \mathbf{y}_t \\
&= h(x) - k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} h_{1:t} - k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} \epsilon_{1:t},
\end{aligned}$$

where  $h_{1:t} = [h(x_1), h(x_2), \dots, h(x_t)]^\top$  and  $\epsilon_{1:t} = [\epsilon_1, \epsilon_2, \dots, \epsilon_t]^\top$ .

We focus on the first term which can be rewritten as

$$\begin{aligned}
|h(x) - k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} h_{1:t}| &= |\phi^\top(x)h - \phi^\top(x)\tilde{A}_t^{-1}\tilde{\Phi}_t^\top h_{1:t}| \\
&= |\phi^\top(x)h - \phi^\top(x)\tilde{A}_t^{-1}\tilde{\Phi}_t^\top \Phi_t h| \\
&\leq \left\| \phi^\top(x) \left( I - \tilde{A}_t^{-1}\tilde{\Phi}_t^\top \Phi_t \right) \right\|_{H_k} \|h\|_{H_k}.
\end{aligned}$$

Consider,

$$\begin{aligned}
\tilde{A}_t^{-1}\tilde{\Phi}_t^\top \Phi_t - I &= \tilde{A}_t^{-1} \left( \tilde{\Phi}_t^\top \Phi_t - \tilde{A}_t \right) \\
&= \tilde{A}_t^{-1} \left( \tilde{\Phi}_t^\top \Phi_t - (\tilde{\Phi}_t^\top \tilde{\Phi}_t + \lambda I) \right) \\
&= \tilde{A}_t^{-1} \left( \tilde{\Phi}_t^\top \Phi_t - \tilde{\Phi}_t^\top \Phi_t P_t - \lambda I \right) \\
&= \tilde{A}_t^{-1} \tilde{\Phi}_t^\top \Phi_t (I - P_t) - \lambda \tilde{A}_t^{-1}.
\end{aligned}$$

Thus,

$$\begin{aligned}
\|\phi^\top(x) \left( I - \tilde{A}_t^{-1} \tilde{\Phi}_t^\top \Phi_t \right)\|_{H_k} &= \|\phi^\top(x) \left( \tilde{A}_t^{-1} \tilde{\Phi}_t^\top \Phi_t (I - P_t) - \lambda I \right)\|_{H_k} \\
&\leq \|\phi^\top(x) \tilde{A}_t^{-1} \tilde{\Phi}_t^\top \Phi_t (I - P_t)\|_{H_k} + \lambda \|\phi^\top(x) \tilde{A}_t^{-1}\|_{H_k} \\
&\leq \sqrt{\phi^\top(x) \tilde{A}_t^{-1} \tilde{\Phi}_t^\top \Phi_t (I - P_t)^2 \Phi_t^\top \tilde{\Phi}_t \tilde{A}_t^{-1} \phi(x)} + \lambda \sqrt{\phi^\top(x) \tilde{A}_t^{-2} \phi(x)} \\
&\leq \sqrt{\phi^\top(x) \tilde{A}_t^{-1} \tilde{\Phi}_t^\top \Phi_t (I - P_t) \Phi_t^\top \tilde{\Phi}_t \tilde{A}_t^{-1} \phi(x)} + \lambda \sqrt{\phi^\top(x) \tilde{A}_t^{-2} \phi(x)} \\
&\leq \sqrt{\frac{\lambda}{1-\varepsilon} \phi^\top(x) \tilde{A}_t^{-1} \tilde{\Phi}_t^\top \tilde{\Phi}_t \tilde{A}_t^{-1} \phi(x)} + \lambda \sqrt{\phi^\top(x) \tilde{A}_t^{-2} \phi(x)} \\
&\leq \sqrt{2 \left\{ \frac{\lambda}{1-\varepsilon} [\phi^\top(x) \tilde{A}_t^{-1} \phi(x) - \lambda \phi^\top(x) \tilde{A}_t^{-2} \phi(x)] + \lambda^2 \phi^\top(x) \tilde{A}_t^{-2} \phi(x) \right\}} \\
&\leq \sqrt{2 \left\{ \frac{\lambda}{1-\varepsilon} \phi^\top(x) \tilde{A}_t^{-1} \phi(x) + \lambda^2 \phi^\top(x) \tilde{A}_t^{-2} \phi(x) \left(1 - \frac{1}{1-\varepsilon}\right) \right\}} \\
&\leq \sqrt{\left\{ \frac{2\lambda}{1-\varepsilon} \phi^\top(x) \tilde{A}_t^{-1} \phi(x) \right\}} \\
&\leq \sqrt{\frac{2\lambda}{1-\varepsilon}} \tilde{\sigma}_t(x).
\end{aligned}$$

The fifth line follows from Lemma 1 in [51] and by noting that for the given choice of  $q_0$ , the dictionaries are  $\varepsilon$ -accurate with a probability of at least  $1 - \delta_0/3$ . On combining everything, we obtain,

$$|h(x) - k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} h_{1:t}| \leq \|h\|_{H_k} \sqrt{\frac{2\lambda}{1-\varepsilon}} \tilde{\sigma}_t(x).$$

We now consider the other term given by  $k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} \epsilon_{1:t}$ . For simplicity, we denote  $k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} := \zeta^\top(x)$ . Using the fact that the components of  $\epsilon_{1:t}$  are independent  $R$ -sub-Gaussian random variables, we have,

$$\mathbb{E} [\exp(\zeta^\top(x) \epsilon_{1:t})] = \exp\left(\frac{R^2}{2} \|\zeta(x)\|^2\right).$$

To bound the RHS, we focus on bounding the norm of the vector  $\zeta(x)$ . We

have,

$$\begin{aligned}
\|\zeta(x)\|^2 &= \zeta^\top(x) \zeta(x) \\
&= k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t} (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} k_{\mathbf{z}_t, x} \\
&= k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} (\lambda k_{ZZ} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t} - \lambda k_{ZZ}) (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} k_{\mathbf{z}_t, x} \\
&= k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} k_{\mathbf{z}_t, x} - \lambda k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{z}_t} (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} k_{\mathbf{z}_t, x} \\
&\leq k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} k_{\mathbf{z}_t, x} \\
&\leq \frac{1}{\lambda} \left( k(x, x) - k_{\mathbf{z}_t, x}^\top K_{\mathbf{z}_t, \mathbf{z}_t}^{-1} k_{\mathbf{z}_t, x} k_{\mathbf{z}_t, x}^\top (K_{\mathbf{z}_t, \mathbf{z}_t} + \lambda^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} k_{\mathbf{z}_t, x} \right) \\
&\leq \tilde{\sigma}_t^2(x).
\end{aligned}$$

Thus, by using Chernoff-Hoeffding inequality, we can conclude that

$$|k_{\mathbf{z}_t, x}^\top (\lambda K_{\mathbf{z}_t, \mathbf{z}_t} + K_{\mathbf{z}_t, \mathbf{x}_t} K_{\mathbf{x}_t, \mathbf{z}_t})^{-1} K_{\mathbf{z}_t, \mathbf{x}_t} \epsilon_{1:t}| \leq R \tilde{\sigma}_t(x) \sqrt{2 \log \left( \frac{1}{\delta} \right)}$$

holds probability at least  $1 - \delta$ . On adding the two terms, we have the required answer.

### E.3 Proof of Theorem 6.5.1

To establish the lower bound on regret for collaborative learning with personalization, we focus on a simple system with two clients and a central server. The main idea in the proof is to construct a difficult instance for learning which enforces collaboration between clients with different objective functions. The construction of this instance of functions in the given RKHS is based on the ideas developed in Scarlett *et al.* [257]. We first describe some preliminaries of constructing functions of interest in the desired RKHS, which is adopted from

the results in [257], and then proceed with the proof of Theorem 6.5.1. Throughout the remaining proof, we assume that the underlying RKHS corresponds to a given kernel belonging to the family of Squared Exponential and Matern kernels.

### E.3.1 Preliminaries on constructing functions in RKHS

Consider the following multi-dimensional bump function defined over  $\mathbb{R}^d$ ,

$$\Phi(\xi) = \begin{cases} \exp\left(-\frac{1}{1-\|\xi\|^2}\right) & \text{if } \|\xi\|^2 \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

Let  $\phi(x)$  be inverse Fourier transform of  $\Phi(\xi)$ . Since  $\Phi$  is a function with finite energy, we can conclude that there exists a  $\zeta > 0$  such that for  $\|x\|_\infty > \zeta$ ,  $\phi(x) \leq 0.5\phi(0)$ . Using  $\phi(x)$ , we define  $\bar{\phi}(x; \theta)$  as the following function for any  $\theta > 0$ :

$$\bar{\phi}(x; \theta, B) = \frac{\theta}{\phi(0)} \phi\left(\frac{x\zeta}{w_\theta}\right),$$

where  $w_\theta$  is a parameter that ensures that the RKHS norm of  $\bar{\phi}$  is at most  $B$ . It is shown in [257] that for any given kernel belonging to the family of Squared Exponential and Matern kernels and constant  $B > 0$ , one can choose  $w_\theta$ , based on  $\theta$ , such that the corresponding RKHS norm of  $\bar{\phi}$  is at most  $B$  for all  $\theta \leq \theta_0(B)$ . In particular, for Square Exponential Kernel  $w_\theta = C (\log(B/\theta))^{-1/2}$  and for Matern Kernel with smoothness parameter  $\nu$ ,  $w_\theta = C'(\theta/B)^{1/\nu}$  for some constants  $C, C' > 0$ . Here  $\theta_0(B)$  is a threshold based on the kernel parameters and  $B$ . For exact expressions of these results, please refer to [257]. For simplicity of notation, we will only consider defining  $\bar{\phi}$  for  $\theta \leq \theta_0$  and implicitly assume the choice of  $w_\theta$  is tuned to the particular  $\theta$ .

Using  $\bar{\phi}$ , we define the following two functions:

$$\varphi_0(x) := \bar{\phi}(x; \theta_1, B_0/2); \quad \varphi_1(x) := \bar{\phi}(x; \varepsilon, B_0/2).$$

In the above definitions,  $\theta_1 = 0.5 \cdot \theta_0(B_0/2)$ ,  $B_0$  is the required upper bound on the observation functions and  $\varepsilon > 0$  is a small constant whose value will be specified later.

### E.3.2 Establishing the lower bound on regret

With this basic tool for constructing RKHS functions, we are ready to build a difficult instance to establish our lower bound. For simplicity of exposition, we consider the case of  $K = 2$  clients. The extension to the general case is straightforward which we briefly describe at the end of the proof. We consider the case of collaborative learning between two clients with observation functions  $h_1$  and  $h_2$  respectively and personalization parameters  $\alpha_1$  and  $\alpha_2$ . We allow the pair  $(\alpha_1, \alpha_2)$  to be any pair in  $[0, 1] \times [0, 1]$  satisfying  $\alpha_\star := \max\{\min\{\alpha_1, 1 - \alpha_1\}, \min\{\alpha_2, 1 - \alpha_2\}\} > 0$ . The objective functions of the two clients are given as  $f_1 = \eta_1 h_1 + (1 - \eta_1)h_2$  and  $f_2 = \eta_2 h_1 + (1 - \eta_2)h_2$ , where  $\eta_1 = (1 + \alpha_1)/2$  and  $\eta_2 = (1 + \alpha_2)/2$ . Consequently,  $(\eta_1, \eta_2) \in \{(x, y) \in [0.5, 1]^2 : \max\{x, y\} \geq 0.5 + \Delta, \min\{x, y\} \leq 1 - \Delta\}$  where  $\Delta = \alpha_\star/2$ .

We begin with considering the cases for which the pair  $(\eta_1, \eta_2)$  satisfies the condition  $\eta_1 \in [1/2, 1 - \Delta]$  and  $\eta_2 \in [1/2 + \Delta, 1]$ . Note that the proof is identical for the pairs of  $(\eta_1, \eta_2)$  satisfying  $\eta_1 \in [1/2 + \Delta, 1]$  and  $\eta_2 \in [1/2, 1 - \Delta]$  by simply exchanging the role of the clients 1 and 2.

To construct the instance of interest, consider the domain given as  $\mathcal{X} = \mathcal{X}_0 \cup \mathcal{X}_1$ , where  $\mathcal{X}_0 = [0, 1]^d$  and  $\mathcal{X}_1$  is a shifted version of  $\mathcal{X}_0$ , centered at  $\bar{z} = (v_0 +$

$3/2, 0, 0, \dots, 0$ ). In the above definition,  $v_0 = \inf\{v : \forall x \text{ s.t. } \|x\|_2 \geq v, \varphi_1(x) \leq 1/T^2\}$ .

The existence of such an  $v_0$  is guaranteed by the fact that  $\phi(x)$  decays to zero faster than any power of  $\|x\|_2$ . See Remark E.3.2 for more details.

To define our observation functions, we consider a collection of  $m_0 = \lfloor w_\varepsilon^{-d} \rfloor$  points, denoted by  $\{z_1, z_2, \dots, z_{m_0}\}$ , which forms a uniform grid over  $\mathcal{X}_0$ . For a given pair  $(\eta_1, \eta_2)$  we define our observation function  $h_1$  and  $h_2$  to be as follows:

$$h_1(x) := -\frac{(1-\eta_1)\Delta}{\eta_1 + \eta_2 - 1} \varphi_0(x - \bar{z}); \quad h_2(x) = \frac{\eta_1\Delta}{\eta_1 + \eta_2 - 1} \varphi_0(x - \bar{z}) + \varphi_1(x - z_M).$$

In the above definition,  $M$  is a random variable chosen uniformly over the set  $\{1, 2, \dots, m_0\}$ . Consequently,  $h_2$  is a random function based on the value of  $M$ . It is not difficult to note that coefficients of  $\varphi_0$  and  $\varphi_1$  in both the functions are less than 1 and hence the RKHS norm of both  $h_1$  and  $h_2$  is less than  $B_0$ , as required. For the observations, we assume that the noise is Gaussian with unit variance.

With this choice of observation functions, our objective functions are given as:

$$f_1(x) := (1-\eta_1)\varphi_1(x - z_M); \quad f_2(x) = \Delta\varphi_1(x) + \eta_2\varphi_1(x - z_M).$$

Note that client 2 does not need to sample any point in  $\mathcal{X}_0$  to optimize its own objective, but would need to do so in order to help client 1 optimize their objective function. This will form the crux of establishing the lower bound.

We introduce some additional notation for the analysis of the lower bound. Throughout the proof, a subscript  $i$  will be used to refer to quantities related to client  $i = 1, 2$ . Let  $\{\mathcal{R}_m\}_{m=1}^{m_0}$  be partition of  $\mathcal{X}_0$  consisting of  $m_0$  regions centred on the points  $\{z_1, z_2, \dots, z_{m_0}\}$ . Let  $(x_{i,t}, y_{i,t})$  denote the input observation pair at time  $t$  and consequently we define  $j_{i,t} = \{m : x_{i,t} \in \mathcal{R}_m\}$ . Using this, we can define  $N_{i,m} = \sum_{t=1}^T \mathbb{1}\{j_{i,t} = m\}$  for  $m \in \{1, 2, \dots, m_0\}$ .

We use  $\mathbb{P}_m(\cdot)$  to denote the distribution of the rewards when  $M = m$  and  $\mathbb{P}_*(\cdot)$  to denote the joint distribution of the rewards and the random variable  $M$ . Let  $\mathbb{P}_0$  denote the reward distribution for the case when  $h_2(x) = \frac{\eta_1 \Delta}{\eta_1 + \eta_2 - 1} \varphi_0(x - \bar{z})$ . We denote the corresponding expectations as  $\mathbb{E}_m[\cdot]$ ,  $\mathbb{E}_*[\cdot]$ , and  $\mathbb{E}_0[\cdot]$ .

Let  $\pi = (\pi_1, \pi_2)$  denote the combined strategy for the two clients. For simplicity, we assume that the policy is deterministic. The arguments can be extended for the case of randomized strategies in a straightforward manner. Furthermore, since there is no constraint on communication, we assume that all the observations are exchanged between the two clients. This assumption is without loss of generality as any policy that communicates only a subset of observations can not do any better than the policy that communicates all the observations. Let  $\mathcal{I}_t = (y_{1,t}, y_{2,t})$  denote information gained at time  $t$  and  $\mathcal{I}^{(t)} = (\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_t)$  denote information state at time  $t$ . Since the policy is deterministic,  $x_{1,t}$  and  $x_{2,t}$  are deterministic functions of  $\mathcal{I}^{(t-1)}$ .

We state the following lemma that will be useful in the analysis.

**Lemma E.3.1.** *Let  $a$  be any function defined on the information state vector  $\mathcal{I}^{(T)}$  whose range is in the bounded interval  $[0, A]$ . Then for any  $m \in \{1, 2, \dots, m_0\}$ , we have,*

$$\mathbb{E}_{\mathbb{P}_m}[a(\mathcal{I}^{(T)})] \leq \mathbb{E}_{\mathbb{P}_0}[a(\mathcal{I}^{(T)})] + A \sqrt{\frac{1}{2T^3} + \sum_{l=1}^{m_0} \mathbb{E}_0[N_{2,l}] D_m^l},$$

where  $D_m^l = \max_{x \in \mathcal{R}_l} D(\mathbb{P}_0(y|x) || \mathbb{P}_m(y|x))$ .

The lemma is an adapted version of Lemma B.1 in [16] and Lemmas 3 and 4 in [257]. The proof of the lemma is provided at the end of the section for completeness.

We first focus on bounding the regret of the first client under the policy  $\pi$

denoted by  $R_{1,\pi}$ . We fix the value of  $M = m$ . To lower bound the expected regret incurred by the first client, we upper bound the expected reward earned by the client under the policy  $\pi$ .

$$\begin{aligned}
\sum_{t=1}^T \mathbb{E}_m[f_1(x_{1,t})] &= \sum_{t=1}^T \mathbb{E}_m \left[ \mathbb{1}\{x_{1,t} \in X_1\} f_1(x_{1,t}) + \sum_{l=1}^{m_0} \mathbb{1}\{x_{1,t} \in \mathcal{R}_l\} f_1(x_{1,t}) \right] \\
&\leq \sum_{t=1}^T \mathbb{E}_m \left[ \mathbb{1}\{x_{1,t} \in X_1\} \frac{1-\eta_1}{T^2} + (1-\eta_1) \sum_{l=1}^{m_0} \mathbb{1}\{x_{1,t} \in \mathcal{R}_l\} \vartheta_m^l \right] \\
&\leq \frac{1-\eta_1}{T} + (1-\eta_1) \sum_{l=1}^{m_0} \sum_{t=1}^T \mathbb{E}_m [\mathbb{1}\{x_{1,t} \in \mathcal{R}_l\}] \vartheta_m^l \\
&\leq \frac{1-\eta_1}{T} + (1-\eta_1) \sum_{l=1}^{m_0} \mathbb{E}_m[N_{1,l}] \vartheta_m^l.
\end{aligned}$$

In the above expression,  $\vartheta_m^l = \max_{x \in \mathcal{R}_l} \varphi_1(x - z_m)$ . On plugging in the result of Lemma E.3.1 for  $N_{1,l}$  in the above expression, we obtain

$$\begin{aligned}
\sum_{t=1}^T \mathbb{E}_m[f_1(x_{1,t})] &\leq \frac{1-\eta_1}{T} + (1-\eta_1) \sum_{l=1}^{m_0} \mathbb{E}_m[N_{1,l}] \vartheta_m^l \\
&\leq \frac{1-\eta_1}{T} + (1-\eta_1) \sum_{l=1}^{m_0} \vartheta_m^l \left( \mathbb{E}_0[N_{1,l}] + T \sqrt{\frac{1}{2T^3} + \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}] D_m^r} \right).
\end{aligned}$$

Averaging over the choice of  $M = m$ , we obtain

$$\begin{aligned}
\frac{1}{m_0} \sum_{m=1}^{m_0} \sum_{t=1}^T \mathbb{E}_m[f_1(x_{1,t})] &\leq \frac{1}{m_0} \sum_{m=1}^{m_0} \frac{1-\eta_1}{T} + \frac{1-\eta_1}{m_0} \sum_{m=1}^{m_0} \sum_{l=1}^{m_0} \vartheta_m^l \left( \mathbb{E}_0[N_{1,l}] + T \sqrt{\frac{1}{2T^3} + \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}] D_m^r} \right) \\
\implies \sum_{t=1}^T \mathbb{E}_*[f_1(x_{1,t})] &\leq \frac{1-\eta_1}{T} + \frac{1-\eta_1}{m_0} \sum_{l=1}^{m_0} \left( \sum_{m=1}^{m_0} \vartheta_m^l \right) \mathbb{E}_0[N_{1,l}] \\
&\quad + \frac{T(1-\eta_1)}{m_0} \sum_{m=1}^{m_0} \left( \sum_{l=1}^{m_0} \vartheta_m^l \right) \sqrt{\frac{1}{2T^3} + \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}] D_m^r}.
\end{aligned}$$

Using the result from Lemma 4 in [257], we can conclude that there exists constants  $C_1, C_2, C_3 > 0$  such that

$$\sum_{m=1}^{m_0} \vartheta_m^l \leq C_1 \varepsilon; \quad \sum_{l=1}^{m_0} \vartheta_m^l \leq C_2 \varepsilon; \quad \sum_{m=1}^{m_0} D_m^r \leq C_3 \varepsilon^2.$$

Using the above relations, we can rewrite the previous equation as

$$\begin{aligned}
\sum_{t=1}^T \mathbb{E}_*[f_1(x_{1,t})] &\leq \frac{1-\eta_1}{T} + \frac{1-\eta_1}{m_0} \sum_{l=1}^{m_0} \left( \sum_{m=1}^{m_0} \vartheta_m^l \right) \mathbb{E}_0[N_{1,l}] \\
&\quad + \frac{T(1-\eta_1)}{m_0} \sum_{m=1}^{m_0} \left( \sum_{l=1}^{m_0} \vartheta_m^l \right) \sqrt{\frac{1}{2T^3} + \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}] D_m^r} \\
&\leq \frac{1-\eta_1}{T} + \frac{C_1(1-\eta_1)\varepsilon}{m_0} \sum_{l=1}^{m_0} \mathbb{E}_0[N_{1,l}] + \frac{C_2 T (1-\eta_1)\varepsilon}{m_0} \sum_{m=1}^{m_0} \sqrt{\frac{1}{2T^3} + \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}] D_m^r} \\
&\leq \frac{1-\eta_1}{T} + \frac{C_1(1-\eta_1)\varepsilon T}{m_0} + C_2 T (1-\eta_1)\varepsilon \sqrt{\frac{1}{m_0} \sum_{m=1}^{m_0} \left( \frac{1}{2T^3} + \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}] D_m^r \right)} \\
&\leq \frac{1-\eta_1}{T} + \frac{C_1(1-\eta_1)\varepsilon T}{m_0} + C_2 T (1-\eta_1)\varepsilon \sqrt{\frac{1}{2T^3} + \frac{1}{m_0} \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}] \left( \sum_{m=1}^{m_0} D_m^r \right)} \\
&\leq \frac{1-\eta_1}{T} + \frac{C_1(1-\eta_1)\varepsilon T}{m_0} + C_2 T (1-\eta_1)\varepsilon \sqrt{\frac{1}{2T^3} + \frac{C_3\varepsilon^2}{m_0} \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}]}.
\end{aligned}$$

In the third step, we used the Jensen's inequality.

Using the upper bound on the reward obtained above, the regret can be lower bounded as

$$\mathbb{E}_*[R_{1,\pi}] \geq (1-\eta_1)\varepsilon T \left( 1 - \frac{1}{\varepsilon T^2} - \frac{C_1}{m_0} - C_2 \sqrt{\frac{1}{2T^3} + \frac{C_3\varepsilon^2 \bar{N}}{m_0}} \right),$$

where  $\bar{N} = \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}]$ .

We now focus on lower bounding the regret incurred by client 2. Note that for sufficiently small  $\varepsilon$ , the maximizer for  $f_2$  clearly lies in  $\mathcal{X}_1$ . As a result, whenever client 2 queries a point in  $\mathcal{X}_0$ , it incurs a regret of at least  $\Delta\theta_1 - \varepsilon \geq \Delta\theta_1/2$ . Thus, for any  $M = m$ , the regret incurred by client 2 can be lower bounded as

$$\mathbb{E}_m[R_{2,\pi}] \geq \frac{\Delta\theta_1}{2} \sum_{r=1}^{m_0} \mathbb{E}_m[N_{2,r}] \geq \frac{\Delta\theta_1}{2} \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}].$$

Consequently,

$$\mathbb{E}_*[R_{2,\pi}] \geq \frac{\Delta\theta_1}{2} \sum_{r=1}^{m_0} \mathbb{E}_0[N_{2,r}] = \frac{\Delta\theta_1 \bar{N}}{2}.$$

Recall that  $m_0$  is a function of  $\varepsilon$  based on the kernel, making the lower bounds on regret of both the clients a function of  $\varepsilon$ . We choose a value of  $\varepsilon$  based on  $T$  to obtain a lower bound on the regret. Firstly, upon constraining  $\varepsilon > 3/T^2$  and choosing a large enough value of  $T$  (or equivalently, a small enough  $\varepsilon$ ), we can ensure that  $C_1/m_0 < 1/3$ . Consequently, we have,

$$\begin{aligned}\mathbb{E}_*[R_{1,\pi}] &\geq (1 - \eta_1)\varepsilon T \left( \frac{1}{3} - C_2 \sqrt{\frac{1}{2T^3} + \frac{C_3\varepsilon^2 N}{m_0}} \right), \\ \mathbb{E}_*[R_{2,\pi}] &\geq \frac{\Delta\theta_1 \bar{N}}{2}.\end{aligned}$$

We set  $\varepsilon = C_4(m_0/\bar{N})^{1/2}$  for an appropriately chosen constant  $C_4 > 0$ , which yields  $\mathbb{E}_*[R_{1,\pi}] = \Omega(T(m_0/\bar{N})^{1/2})$  and  $\mathbb{E}_*[R_{2,\pi}] = \Omega(\bar{N})$ . Upon equating the two expressions to ensure the tightest bound, we obtain that  $\bar{N} = \Theta(T^{2/3}m_0^{1/3}) \implies \varepsilon = \Theta((m_0/T)^{1/3})$ .

Plugging in the relation between  $m_0$  and  $\varepsilon$  based on the kernel parameters, we arrive at result. In particular, for a Matern Kernel with smoothness parameter  $\nu$ ,  $m_0 = \Theta(\varepsilon^{-d/\nu})$ , which gives us a values of  $\varepsilon = \Theta(T^{-\nu/(3\nu+d)})$  and consequently a regret lower bound of  $\Omega(\alpha_\star T^{(2\nu+d)/(3\nu+d)})$ . Similarly, for the Squared Exponential Kernel, repeating a similar process yields a lower bound of  $\Omega(\alpha_\star T^{2/3}(\log(T))^{d/6})$ . The leading constant  $\alpha_\star$  follows directly from the lower bounds  $\mathbb{E}_*[R_{1,\pi}]$  and  $\mathbb{E}_*[R_{2,\pi}]$  where  $1 - \eta_1 \geq \Delta = \alpha_\star/2$ .

*Remark E.3.2. Choosing the value of  $v_0$ :* Note that for the final choice of  $\varepsilon$ , the corresponding parameter  $w_\varepsilon = \Theta(T^{-1/(3\nu+d)})$ . Since the function  $\phi(x)$  decays to zero faster than any power of  $\|x\|_2$ , there exists constants  $C_5, C_6$  such that for all  $x$  with  $\|x\|_2 > C_5$ ,  $\phi(x) \leq C_6/\|x\|_2^{2(3\nu+d)}$ . Consequently, for any  $\nu > C_5$ ,  $\varphi_1(x) \leq 1/T^2$  due to the scaling with  $w_\varepsilon$ . Thus, one can guarantee  $v_0$  to be finite and it can be chosen based on the decay of  $\phi(x)$  and the kernel parameters.

### E.3.3 Extension to the case of $K > 2$

For the case of  $K > 2$ , the approach is very similar to the case of  $K = 2$ . The only difference is in the construction of the observation and objective functions.

WLOG, let  $\alpha_1 \leq \alpha_2 \leq \dots \leq \alpha_K$ . Similar to the case of  $K = 2$ , let  $\eta_i = (1 + (K - 1)\alpha_i)/K$  and  $f_i = \eta_i h_i + \frac{1 - \eta_i}{K - 1} \sum_{k=1, k \neq i}^K h_k$ . The observation functions are  $h_1, h_2, \dots, h_K$  are defined as follows:

$$\begin{aligned} h_1 &= -\frac{(1 - \eta_1)\lambda_0}{K - 1}\varphi_0(x - \bar{z}); \\ h_i &= -\frac{\eta_1\lambda_0}{K - 1}\varphi_0(x - \bar{z}); \quad \forall i = \{2, 3, \dots, K - 1\}, \\ h_K &= \eta_1(K - 1)\lambda_0\varphi_0(x - \bar{z}) + \varphi_1(x - z_M), \end{aligned}$$

where  $\bar{z}$  and  $z_M$  are defined as before and  $\lambda_0 := \frac{\Delta'(K - 1)^2}{\eta_1\eta_K(K^3 - 2K^2) + \eta_K - \eta_1(K^2 - 3K + 1) - 1}$  with  $\Delta' := (K - 1)\alpha_\star/K$ .

With this choice of the observation functions, the objective function of client  $i = \{1, 2, \dots, K - 1\}$  is given by

$$f_i = \left( \frac{\eta_i + (2K - 1)\eta_1 - K^2\eta_i\eta_1 - 1}{(K - 1)^2} \right) \lambda_0\varphi_0(x - \bar{z}) + \frac{1 - \eta_i}{K - 1}\varphi_1(x - z_M).$$

Since  $\eta_i \geq \eta_1$  (because  $\alpha_i \geq \alpha_1$ ),  $\frac{\eta_i + (2K - 1)\eta_1 - K^2\eta_i\eta_1 - 1}{(K - 1)^2} \leq -\left(\frac{\eta_1 K - 1}{K - 1}\right)^2 \leq 0$ .

This implies that the maximizer of  $f_i$  lies in  $\mathcal{X}_0$ . On the other hand, the objective function of client  $K$  is given as

$$f_K = \Delta'\varphi_0(x - \bar{z}) + \frac{1 - \eta_K}{K - 1}\varphi_1(x - z_M).$$

Similar to the case of  $K = 2$ , the maximizer of  $f_K$  lies in  $\mathcal{X}_1$  but it would need to sample points in  $\mathcal{X}_0$  in order to help other clients. The rest of the argument follows the same way as for the case of  $K = 2$  leading to the same conclusion.

### E.3.4 Proof of Lemma E.3.1

The proof of this lemma is similar to the lemma obtained for the lower bound in adversarial bandits in Auer *et al.* [16]. We have,

$$\begin{aligned}
\mathbb{E}_m[a(\mathcal{I}^{(T)})] - \mathbb{E}_0[a(\mathcal{I}^{(T)})] &= \sum_{\mathcal{I}^{(T)}} a(\mathcal{I}^{(T)}) (\mathbb{P}_m(\mathcal{I}^{(T)}) - \mathbb{P}_0(\mathcal{I}^{(T)})) \\
&\leq \sum_{\mathcal{I}^{(T)}: \mathbb{P}_m(\mathcal{I}^{(T)}) \geq \mathbb{P}_0(\mathcal{I}^{(T)})} a(\mathcal{I}^{(T)}) (\mathbb{P}_m(\mathcal{I}^{(T)}) - \mathbb{P}_0(\mathcal{I}^{(T)})) \\
&\leq A \sum_{\mathcal{I}^{(T)}: \mathbb{P}_m(\mathcal{I}^{(T)}) \geq \mathbb{P}_0(\mathcal{I}^{(T)})} (\mathbb{P}_m(\mathcal{I}^{(T)}) - \mathbb{P}_0(\mathcal{I}^{(T)})) \\
&\leq \frac{A}{2} \|\mathbb{P}_m(\mathcal{I}^{(T)}) - \mathbb{P}_0(\mathcal{I}^{(T)})\|_1 \\
&\leq A \sqrt{D(\mathbb{P}_0(\mathcal{I}^{(T)}) || \mathbb{P}_m(\mathcal{I}^{(T)}))}.
\end{aligned}$$

In the last step, we have used Pinsker's inequality and  $D(\mathbb{P} || \mathbb{Q})$  denotes the Kullback Leibler Divergence between two distributions  $\mathbb{P}$  and  $\mathbb{Q}$ .

We can compute the KL divergence between  $\mathbb{P}_0$  and  $\mathbb{P}_m$  using the chain rule. We have,

$$\begin{aligned}
D(\mathbb{P}_0 || \mathbb{P}_m) &= \sum_{t=1}^T D(\mathbb{P}_0(\mathcal{I}_t | \mathcal{I}^{(t-1)}) || \mathbb{P}_m(\mathcal{I}_t | \mathcal{I}^{(t-1)})) \\
&= \sum_{t=1}^T D(\mathbb{P}_0((y_{1,t}, y_{2,t}) | \mathcal{I}^{(t-1)}) || \mathbb{P}_m((y_{1,t}, y_{2,t}) | \mathcal{I}^{(t-1)})) \\
&= \sum_{t=1}^T D(\mathbb{P}_0(y_{1,t} | \mathcal{I}^{(t-1)}) || \mathbb{P}_m(y_{1,t} | \mathcal{I}^{(t-1)})) + D(\mathbb{P}_0(y_{2,t} | \mathcal{I}^{(t-1)}) || \mathbb{P}_m(y_{2,t} | \mathcal{I}^{(t-1)})) \\
&= \sum_{t=1}^T D(\mathbb{P}_0(y_{2,t} | \mathcal{I}^{(t-1)}) || \mathbb{P}_m(y_{2,t} | \mathcal{I}^{(t-1)})) \\
&= \sum_{t=1}^T \mathbb{E}_0 \left[ \left( \mathbb{1}\{x_{2,t} \in \mathcal{X}_1\} + \sum_{l=1}^{m_0} \mathbb{1}\{x_{2,t} \in \mathcal{R}_l\} \right) D(\mathbb{P}_0(y_{2,t} | x_{2,t}) || \mathbb{P}_m(y_{2,t} | x_{2,t})) \right].
\end{aligned}$$

The third step uses the independence of observations across clients (conditioned on the query points) and the fourth step follows by noting that reward distribu-

tion of client 1 is the same under both  $\mathbb{P}_0$  and  $\mathbb{P}_m$ . Since  $\mathbb{P}_0$  and  $\mathbb{P}_m$  are Gaussian distributions with unit variance, their KL divergence is half the square of difference of their means. Using the definition of  $v_0$ , we can conclude that  $D(\mathbb{P}_0(y_{2,t}|x_{2,t})||\mathbb{P}_m(y_{2,t}|x_{2,t})) \leq 1/(2T^4)$  for all  $x_{2,t} \in \mathcal{X}_1$ . Using the definition of  $D_m^l$  for the regions in  $\mathcal{X}_0$ , we can rewrite the previous equation as,

$$\begin{aligned}
D(\mathbb{P}_0||\mathbb{P}_m) &= \sum_{t=1}^T \mathbb{E}_0 \left[ \left( \mathbb{1}\{x_{2,t} \in \mathcal{X}_1\} + \sum_{l=1}^{m_0} \mathbb{1}\{x_{2,t} \in \mathcal{R}_l\} \right) D(\mathbb{P}_0(y_{2,t}|x_{2,t})||\mathbb{P}_m(y_{2,t}|x_{2,t})) \right] \\
&\leq \sum_{t=1}^T \mathbb{E}_0 \left[ \mathbb{1}\{x_{2,t} \in \mathcal{X}_1\} \cdot \frac{1}{2T^2} + \sum_{l=1}^{m_0} \mathbb{1}\{x_{2,t} \in \mathcal{R}_l\} D_m^l \right] \\
&\leq T \cdot \frac{1}{2T^4} + \sum_{l=1}^{m_0} \mathbb{E}_0 \left[ \sum_{t=1}^T \mathbb{1}\{x_{2,t} \in \mathcal{R}_l\} \right] D_m^l \\
&\leq \frac{1}{2T^3} + \sum_{l=1}^{m_0} \mathbb{E}_0[N_{2,l}] D_m^l.
\end{aligned}$$

APPENDIX F  
CHAPTER 7

### F.1 Proof of Theorem 7.4.1

We first introduce the following normalized uniform convergence VC bound from the result in Vapnik and Chervonenkis [309].

**Lemma F.1.1.** *Let  $\mathcal{F}$  be a family of measurable functions  $f : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ . Let  $\mathbb{Q}$  be a fixed distribution over  $\mathcal{X} \times \mathcal{Y}$ . Define*

$$\mathbb{Q}f = \mathbb{E}_{X,Y \sim \mathbb{Q}} f(X, Y). \quad (\text{F.1})$$

For a finite set  $\mathcal{Z} \subseteq \mathcal{X} \times \mathcal{Y}$ , define

$$\mathbb{Q}_{\mathcal{Z}}f = \frac{1}{|\mathcal{Z}|} \sum_{(X,Y) \in \mathcal{Z}} f(X, Y) \quad (\text{F.2})$$

as the empirical average of  $f$  over  $\mathcal{Z}$ . If  $\mathcal{Z}$  is an i.i.d. sample of size  $n$  from  $\mathbb{Q}$ , then, with probability at least  $1 - \delta$ , for all  $f \in \mathcal{F}$ :

$$\gamma_n \sqrt{\mathbb{Q}_{\mathcal{Z}}f} \leq \mathbb{Q}f - \mathbb{Q}_{\mathcal{Z}}f \leq \gamma_n^2 + \gamma_n \sqrt{\mathbb{Q}_{\mathcal{Z}}f}, \quad (\text{F.3})$$

where  $\gamma_n = \sqrt{(4/n) \ln(8\mathcal{S}(\mathcal{F}, 2n)/\delta)}$ .

Define

$$g_{h_1, h_2}(x, y) := \mathbb{1}[h_1(x) \neq y \wedge h_2(x) = y], \quad (\text{F.4})$$

which is a measurable mapping from  $\mathcal{X} \times \mathcal{Y}$  to  $\{0, 1\}$ . It is straightforward to note that  $\mathbb{P}(g_{h_1, h_2}) = \epsilon_{\mathbb{P}}(h_1, h_2)$  and  $\mathbb{P}_{\mathcal{Z}}(g_{h_1, h_2}) = \epsilon_{\mathcal{Z}}(h_1, h_2)$ .

On applying the normalized VC bound (Lemma F.1.1) to the family of measurable functions  $\mathcal{F} = \{g_{h_1, h_2} \mid h_1, h_2 \in \mathcal{H}\} = \{g_{h_2, h_1} \mid h_1, h_2 \in \mathcal{H}\}$  defined in Eqn. (F.4), we obtain the following two inequalities:

$$\epsilon_{\mathbb{P}}(h_1, h_2) - \epsilon_{\mathcal{Z}}(h_1, h_2) \leq \gamma_n^2 + \gamma_n \sqrt{\epsilon_{\mathcal{Z}}(h_1, h_2)}, \quad (\text{F.5})$$

$$\epsilon_{\mathcal{Z}}(h_2, h_1) - \epsilon_{\mathbb{P}}(h_2, h_1) \leq \gamma_n \sqrt{\epsilon_{\mathcal{Z}}(h_2, h_1)}. \quad (\text{F.6})$$

Note that we can relate  $g_{h_1, h_2}(x, y)$  and  $g_{h_2, h_1}(x, y)$  for any  $h_1, h_2 \in \mathcal{H}$  as follows:

$$\begin{aligned} \mathbb{1}[h_1(x) \neq y] - \mathbb{1}[h_2(x) \neq y] &= \mathbb{1}[h_1(x) \neq y \wedge h_2(x) = y] - \mathbb{1}[h_2(x) \neq y \wedge h_1(x) = y] \\ &= g_{h_1, h_2}(x, y) - g_{h_2, h_1}(x, y). \end{aligned}$$

On taking expectation w.r.t.  $\mathbb{P}$  and  $\mathbb{P}_{\mathcal{Z}}$ , we can conclude that  $\epsilon_{\mathbb{P}}(h_1) - \epsilon_{\mathbb{P}}(h_2) = \epsilon_{\mathbb{P}}(h_1, h_2) - \epsilon_{\mathbb{P}}(h_2, h_1)$  and  $\epsilon_{\mathcal{Z}}(h_1) - \epsilon_{\mathcal{Z}}(h_2) = \epsilon_{\mathcal{Z}}(h_1, h_2) - \epsilon_{\mathcal{Z}}(h_2, h_1)$ . Upon adding Eqn. (F.5) and Eqn. (F.6) and using above relations, we arrive at the required relation.

## F.2 Proof of Theorem 7.4.3

We first show that  $h^* \in \mathcal{H}_k$  for all  $k \geq 1$ . This claim follows using an inductive application of Corollary 7.4.2. For the base case, note that  $h^* \in \mathcal{H}_1$  holds by definition. For the inductive step, assume that  $h^* \in \mathcal{H}_k$ . On invoking Corollary 7.4.2 with  $h = h^*$ , we obtain,

$$\begin{aligned} \epsilon_{\mathbb{P}|\mathcal{D}_k}(h_k^*) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h^*) &\leq \epsilon_{\mathcal{Z}_k}(h_k^*) - \epsilon_{\mathcal{Z}_k}(h^*) + \Delta_{\mathcal{Z}_k}(h, h_k^*) \\ \implies \epsilon_{\mathcal{Z}_k}(h^*) - \epsilon_{\mathcal{Z}_k}(h_k^*) &\leq \Delta_{\mathcal{Z}_k}(h, h_k^*) + \epsilon_{\mathbb{P}|\mathcal{D}_k}(h^*) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h_k^*) \\ &\leq \Delta_{\mathcal{Z}_k}(h, h_k^*). \end{aligned}$$

The last step follows by noting that  $\epsilon_{\mathbb{P}|\mathcal{D}_k}(h^*) \leq \epsilon_{\mathbb{P}|\mathcal{D}_k}(h_k^*)$ .

From the querying rule of OLA, we can conclude that  $h^* \in \mathcal{H}_{k+1}$ , completing the inductive step. Since Corollary 7.4.2 holds for all  $k \geq 1$  with probability at

least  $1 - \frac{1}{2T}$ , we have  $\Pr(h^* \in \mathcal{H}_k, \forall k \geq 1) \geq 1 - \frac{1}{2T}$  which implies  $\Pr(R(T) = 0) \geq 1 - \frac{1}{2T}$ . The implication follows by noting that the labeling rule of OLA ensures that whenever  $h^* \in \mathcal{H}_k$ , the algorithm incurs no regret. Consequently,

$$\mathbb{E}[R(T)] \leq \Pr(R(T) > 0) \cdot T = \frac{1}{2T} \cdot T = 1/2. \quad (\text{F.7})$$

as desired.

### F.3 Proof of Theorem 7.4.4

Let  $k_t$  be epoch index at time  $t$  and let  $\mathcal{A} = \{t < T : \phi(\mathcal{D}_{k_t}) < T^{-\frac{\alpha}{2-\alpha}}\}$ . We define  $\tau := (\min \mathcal{A}) - 1$  whenever  $\mathcal{A} \neq \emptyset$ , and  $\tau := T$ , otherwise. We separate the analysis into two stages:  $t \leq \tau$  and  $t > \tau$ . We begin with bounding the label complexity of the first stage,  $\mathbb{E}[Q(\tau)]$ . Let  $c = 18^\alpha \theta c_0 m^{-\frac{\alpha}{2}} < 1$ . With this choice of  $c$  and using the definition of  $\beta_M$ , we can conclude that,  $3\beta_M^2 + 3\sqrt{2}\beta_M \leq 18/\sqrt{mT^{\frac{2-2\alpha}{2-\alpha}}}$ . Let  $\mathcal{H}_k^\theta = \left\{ h \in \mathcal{H}_k, \rho(h, h^*) > \frac{c\phi(\mathcal{D}_k)}{\theta} \right\}$ . For any  $h \in \mathcal{H}_k^\theta$ , we have,

$$\epsilon_{\mathbb{P}|\mathcal{D}_k}(h) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h^*) = \frac{\epsilon_{\mathbb{P}}(h) - \epsilon_{\mathbb{P}}(h^*)}{\phi(\mathcal{D}_k)} \geq \left( \frac{\rho(h, h^*)}{c_0} \right)^{\frac{1}{\alpha}} \frac{1}{\phi(\mathcal{D}_k)}.$$

Since  $\phi(\mathcal{D}_k) < T^{-\frac{\alpha}{2-\alpha}}$  for all  $k \leq k_\tau$ , we have

$$\left( \frac{\rho(h, h^*)}{c_0} \right)^{\frac{1}{\alpha}} \frac{1}{\phi(\mathcal{D}_k)} = \frac{18\phi(\mathcal{D}_k)^{\frac{1}{\alpha}}}{\sqrt{m}} \frac{1}{\phi(\mathcal{D}_k)} \geq \frac{18}{\sqrt{mT^{\frac{2-2\alpha}{2-\alpha}}}}.$$

Thus by Corollary 7.4.2, we can conclude that

$$\begin{aligned}
\epsilon_{\mathcal{Z}_k}(h) - \epsilon_{\mathcal{Z}_k}(h_k^*) &\geq \epsilon_{\mathbb{P}|\mathcal{D}_k}(h) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h_k^*) - \beta_M^2 - \beta_M \left( \sqrt{\epsilon_{\mathcal{Z}_k}(h, h_k^*)} + \sqrt{\epsilon_{\mathcal{Z}_k}(h_k^*, h)} \right) \\
&\geq \epsilon_{\mathbb{P}|\mathcal{D}_k}(h) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h^*) + \epsilon_{\mathbb{P}|\mathcal{D}_k}(h^*) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h_k^*) - \beta_M^2 - \sqrt{2}\beta_M \\
&> \frac{18}{\sqrt{mT^{\frac{2-2\alpha}{2-\alpha}}}} - \Delta_{\mathcal{Z}_k}(h^*, h_k^*) - \beta_M^2 - \sqrt{2}\beta_M \\
&> \frac{18}{\sqrt{mT^{\frac{2-2\alpha}{2-\alpha}}}} - 2\beta_M^2 - 2\sqrt{2}\beta_M \\
&> \beta_M^2 + \sqrt{2}\beta_M \\
&> \Delta_{\mathcal{Z}_k}(h, h_k^*)
\end{aligned}$$

holds with probability at least  $1 - \frac{1}{2T}$ . This implies that for all  $h \in \mathcal{H}_k^\theta$ ,  $h \notin \mathcal{H}_{k+1}$ . Consequently, using the definition of  $\theta$ , we can conclude that the following relation holds with probability at least  $1 - 1/(2T)$ :  $\phi(\mathcal{D}_{k+1}) \leq \phi(\Psi(\mathcal{H}_k \setminus \mathcal{H}_k^\theta)) \leq \frac{c\phi(\mathcal{D}_k)}{\theta} \cdot \theta = c\phi(\mathcal{D}_k)$ . Therefore,  $\mathbb{E}[\phi(\mathcal{D}_{k+1})|\phi(\mathcal{D}_k)] \leq (\frac{1+c}{2})\phi(\mathcal{D}_k)$ . On applying this repeatedly, we obtain the following relation.

$$\mathbb{E}[\phi(\mathcal{D}_k)] \leq \left(\frac{1+c}{2}\right)^k \phi(\mathcal{D}_0) = \left(\frac{1+c}{2}\right)^k.$$

Define  $S(t) = \left(\frac{2}{1+c}\right)^{\frac{Q(t)}{M}} - \left(\frac{2}{1+c}\right) \left[ \left(\frac{2}{1+c}\right)^{\frac{1}{M}} - 1 \right] t$ , where recall that  $Q(t)$  denotes the label complexity at time  $t$ . We show that  $S_t$  is a supermartingale which will help us bound its expected value using the expected value of  $S(0)$ . We have,

$$\begin{aligned}
\Pr(Q(t+1) = Q(t) + 1 | S(1), S(2), \dots, S(t)) &= \Pr(q_{t+1} = 1 | S(1), S(2), \dots, S(t)) \\
&= \mathbb{E}[\phi(\mathcal{D}_{k_t})|Q(t)] \\
&= \mathbb{E}[\phi(\mathcal{D}_{\lfloor \frac{Q(t)}{M} \rfloor})] \\
&\leq \left(\frac{1+c}{2}\right)^{\frac{Q(t)}{M}-1}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathbb{E}[S(t+1)|S_1, \dots, S(t)] &= \mathbb{E}\left[\left(\frac{2}{1+c}\right)^{\frac{Q(t)}{M}} - \left(\frac{2}{1+c}\right)\left(\left(\frac{2}{1+c}\right)^{\frac{1}{M}} - 1\right)t \middle| S_1, \dots, S(t)\right] \\
&\leq \left(\frac{2}{1+c}\right)^{\frac{Q(t)+1}{M}} \left(\frac{1+c}{2}\right)^{\frac{Q(t)}{M}-1} + \left(\frac{2}{1+c}\right)^{\frac{Q(t)}{M}} \left(1 - \left(\frac{1+c}{2}\right)^{\frac{Q(t)-1}{M}}\right) \\
&\quad - \left(\frac{2}{1+c}\right)\left[\left(\frac{2}{1+c}\right)^{\frac{1}{M}} - 1\right](t+1) \\
&= \left(\frac{2}{1+c}\right)^{\frac{Q(t)}{M}} - \left(\frac{2}{1+c}\right)\left[\left(\frac{2}{1+c}\right)^{\frac{1}{M}} - 1\right]t \\
&= S(t),
\end{aligned}$$

implying  $S(t)$  is a supermartingale. Then by optional stopping theorem, we have,

$$\mathbb{E}\left[\left(\frac{2}{1+c}\right)^{\frac{Q(\tau)}{M}} - \left(\frac{2}{1+c}\right)\left(\left(\frac{2}{1+c}\right)^{\frac{1}{M}} - 1\right)\tau\right] = \mathbb{E}[S(\tau)] \leq \mathbb{E}[S(0)] = 1$$

Since  $\tau \leq T$ , we have,

$$\begin{aligned}
\mathbb{E}\left[\left(\frac{2}{1+c}\right)^{\frac{Q(\tau)}{M}}\right] &\leq \left(\frac{2}{1+c}\right)\left(\left(\frac{2}{1+c}\right)^{\frac{1}{M}} - 1\right)T + 1 \\
\implies \mathbb{E}[Q(\tau)] &\leq M \log_{\frac{2}{1+c}}\left[\left(\frac{2}{1+c}\right)\left(\left(\frac{2}{1+c}\right)^{\frac{1}{M}} - 1\right)T + 1\right].
\end{aligned}$$

where the second relation follows using Jensen's inequality.

Since  $M \leq mdT^{\frac{2-2\alpha}{2-\alpha}} \log T + 1$ ,  $c < 1$  and  $(\frac{2}{1+c})((\frac{2}{1+c})^{\frac{1}{M}} - 1) < 2$ , we have

$$\mathbb{E}[Q(\tau)] \leq \frac{2mdT^{\frac{2-2\alpha}{2-\alpha}}}{\log \frac{2}{1+c}} (\log T + 1)^2 = O(dT^{\frac{2-2\alpha}{2-\alpha}} \log^2 T)$$

By definition of  $\tau$  we have  $q_t \leq T^{-\frac{\alpha}{2-\alpha}}$ ,  $\forall t > \tau$ . Therefore,

$$\begin{aligned}
\mathbb{E}[Q(T)] &= \mathbb{E}[Q(\tau)] + \mathbb{E}\left[\sum_{t=\tau+1} q_t \middle| \tau\right] \\
&\leq \frac{2mdT^{\frac{2-2\alpha}{2-\alpha}}}{\log \frac{2}{1+c}} (\log T + 1)^2 + T^{-\frac{\alpha}{2-\alpha}} \cdot T \\
&\leq \left(\frac{2md}{\log \frac{2}{1+c}} + 1\right) T^{\frac{2-2\alpha}{2-\alpha}} (\log T + 1)^2,
\end{aligned} \tag{F.8}$$

as desired.

## F.4 Proof of Theorem 7.4.5

For ease of understanding we present the main proof for the case when the instance space,  $\mathcal{X}$ , is the interval  $[0, 1]$  and the hypothesis space,  $\mathcal{H}$ , is the class of threshold classifiers. The extension to general hypothesis spaces is straightforward and is described at the end of the proof. We begin the proof by considering the expression for regret for a (possibly randomized) policy  $\pi$ ,

$$\begin{aligned}
\mathbb{E}[R_\pi(T)] &= \mathbb{E} \left[ \sum_{t=1}^T \mathbb{1}\{q_t = 0\} (\mathbb{1}\{\lambda_t \neq Y_t\} - \mathbb{1}\{h^*(X_t) \neq Y_t\}) \right] \\
&= \mathbb{E} \left[ \sum_{t=1}^T \mathbb{1}\{q_t = 0\} \mathbb{E}_{Y_t} [(\mathbb{1}\{\lambda_t \neq Y_t\} - \mathbb{1}\{h^*(X_t) \neq Y_t\})] \right] \\
&= \mathbb{E} \left[ \sum_{t=1}^T \mathbb{1}\{q_t = 0\} 2 \left| \eta(X_t) - \frac{1}{2} \right| \mathbb{1}\{\lambda_t \neq h^*(X_t)\} \right] \\
&= 2 \sum_{t=1}^T \mathbb{E} \left[ \left| \eta(X_t) - \frac{1}{2} \right| \Pr(q_t = 0 | I_{t-1}) \Pr(\lambda_t \neq h^*(X_t) | I_{t-1}, q_t = 0) \right]
\end{aligned} \tag{F.9}$$

where  $I_t$  denotes the information vector up to time  $t$ . It contains all the information obtained up to and including time  $t$  in terms of observed instances and their labels along with any additional information pertaining to the algorithm up to time  $t$ . We now focus on bounding the probability of making an error.

To bound the probability of making a mistake, we restrict ourselves to a subproblem which is easier to analyze. Fix a  $\mu > 0$  and consider a pair of hypotheses  $h_1, h_2$  such that  $\rho(h_1, h^*) = \mu$ ,  $\rho(h_2, h^*) = \mu/2$  and  $\rho(h_1, h_2) = \mu/2$ . For example, when the distribution is uniform and if a hypothesis is represented by a point in the interval, then a possible option is  $h_1 = h^* - \mu$  and  $h_2 = h^* - \mu/2$ . For the analysis, we would only consider the regret incurred by  $\pi$  in the region  $\mathcal{D}(h_1, h_2)$

where  $\mathcal{D}(h, h') = \{x : h(x) \neq h'(x)\}$ . This is a subset of the instance space and consequently the total regret incurred by  $\pi$  would be at least the regret it incurs on the instances in this region.

To lower bound the probability of error when policy  $\pi$  labels a point, we consider the subproblem of distinguishing between  $h_1$  and  $h^*$ . More specifically, given certain number of labeled instances in  $\mathcal{D}(h_1, h^*)$  we bound the probability that *any* randomized policy would be unable to identify the correct hypothesis between  $h_1$  and  $h^*$  using the labeled instances. The motivation is that if  $\pi$  incorrectly concludes that  $h_1$  is the true hypothesis and proceeds to label a point in  $\mathcal{D}(h_1, h_2)$  then such a labeling event would contribute to regret. And since  $\pi$  is a policy that performs uniformly well for all  $\mathbb{P}_{Y|X}$ , it would have to distinguish between  $h_1$  and  $h^*$  during the course of learning and therefore the regret incurred by the policy is at least as much as it incurs in trying to distinguish these two hypotheses.

Since we are focusing on the binary hypothesis problem of distinguishing between  $h_1$  and  $h^*$ , we can restrict ourselves to the instances observed in  $\mathcal{D}(h_1, h^*)$ . Recall that we are considering threshold classifiers and therefore in the disagreement region between two hypotheses, one of them will label all the points to be  $+1$  while the other would label all of them to be  $-1$ . Consequently, the problem of identifying the correct hypothesis between  $h_1$  and  $h^*$  is equivalent to the problem of identifying the parameter of a Bernoulli random variable. The following technical lemma from Anthony and Bartlett [12], which provides a lower bound on error in estimating the parameter of a Bernoulli random variable, would be useful in further analysis.

**Lemma F.4.1.** *Suppose that  $\alpha$  is a random variable that is uniformly distributed on*

$\{\alpha_+, \alpha_-\}$  where  $\alpha_- = \frac{1}{2} - \frac{\gamma}{2}$  and  $\alpha_+ = \frac{1}{2} + \frac{\gamma}{2}$  with  $\gamma \in (0, 1)$ . Suppose that  $(\xi_1, \xi_2, \dots, \xi_m)$  are i.i.d.  $\{0, 1\}$  random variables with  $\Pr(\xi_i = 1) = \alpha$  for all  $i$ . Let  $f$  be a function, possibly randomized, from  $\{0, 1\}^m \rightarrow \{\alpha_+, \alpha_-\}$ , then we have

$$\begin{aligned}\Pr(f(\xi_1, \xi_2, \dots, \xi_m) \neq \alpha) &> \frac{1}{4} \left( 1 - \sqrt{1 - \exp\left(\frac{-2[m/2]\gamma^2}{1 - \gamma^2}\right)} \right) \\ &> \frac{1}{8} \exp\left(\frac{-2[m/2]\gamma^2}{1 - \gamma^2}\right)\end{aligned}$$

We cannot directly apply lemma since our setup is slightly different from the one in the lemma. It is not difficult to see that in our case, the labels observed are independent but not identically distributed. However, it is straightforward to slightly tweak the proof of the above lemma to incorporate this condition. Let  $m_t$  denote the number of instances queried by the policy  $\pi$  up to (but not including) time instant  $t$  in  $\mathcal{D}(h_1, h^*)$  corresponding to points  $X_1, X_2, \dots, X_{m_t}$ . WLOG we can assume that  $h^*$  labels these points as  $-1$ . The proof of the above lemma argues that probability of error is at least the probability of observing greater than  $m_t/2$  points labeled  $+1$ . The probability of this event is more than that of observing greater than  $m_t/2$  instances of  $+1$  in an i.i.d. sample of size  $m_t$  of a Bernoulli random variable  $Z$ , where  $\Pr(Z = +1) = 1/2 - \gamma_{\max}/2$  where  $\gamma_{\max} = 2 \max_{x \in \mathcal{D}(h_1, h^*)} |\eta(x) - 1/2|$ . This follows from the fact that  $\eta(x) \geq 1/2 - \gamma_{\max}/2$  for all  $x \in \mathcal{D}(h_1, h^*)$ . Therefore, we can lower bound the probability of interest by bounding the probability of error in estimating the parameter for  $Z$  for which we can directly use the lemma. Using the result from the lemma and plugging it back into Eqn. (F.9), we get,

$$\mathbb{E}[R_\pi(T)] \geq 2 \sum_{t=1}^T \mathbb{E} \left[ \left| \eta(X_t) - \frac{1}{2} \right| \Pr(q_t = 0 | I_{t-1}) \mathbb{1}\{X_t \in \mathcal{D}(h_1, h_2)\} \exp\left(\frac{-2[m_t/2]\gamma_{\max}^2}{1 - \gamma_{\max}^2}\right) \right] \quad (\text{F.10})$$

From the assumptions on the noise model, we can conclude that  $\min_{x \in \mathcal{D}(h_1, h_2)} |\eta(x) - 1/2| \geq 0.5c'_2\mu^{1/\alpha-1}$  for some universal constant  $c'_2 > 0$ . Also, if  $Q(T)$  is random variable corresponding to the total number of queries by the policy  $\pi$  until time  $T$ , then we have the trivial inequality  $m_t \leq Q(T)$ . Using these observations, we can rewrite Eqn. (F.10) as,

$$\mathbb{E}[R_\pi(T)] \geq \frac{c'_2}{8}\mu^{1/\alpha-1}\mathbb{E}\left[\sum_{t=1}^T \mathbb{1}\{q_t = 0, X_t \in \mathcal{D}(h_1, h_2)\} \exp\left(\frac{-2[Q(T)/2]\gamma_{\max}^2}{1 - \gamma_{\max}^2}\right)\right] \quad (\text{F.11})$$

To further bound the expression on RHS, we consider the number of instances that arrive in the  $\mathcal{D}(h_1, h_2)$ . Let  $N$  be the random number of observed instances which belong to  $\mathcal{D}(h_1, h_2)$ . To bound the above expression we consider the event  $N \geq \mu T/2$ . From Theorem 1 in [117], we can conclude that the probability of such an event is at least  $1/4$ . To evaluate the expression involving  $Q(T)$ , we consider the event such that  $Q(T) \leq \gamma_{\max}^{-2}$ . Under this event, it is not difficult to note that the following holds  $\exp\left(\frac{-2[Q(T)/2]\gamma_{\max}^2}{1 - \gamma_{\max}^2}\right) \geq c_3$  for a universal constant  $c_3 > 0$ . Combining these two points with Eqn. (F.11), we get,

$$\mathbb{E}[R_\pi(T)] \geq \frac{c'_2}{8}\mu^{1/\alpha-1}\mathbb{E}\left[(N - Q(T)) \exp\left(\frac{-2[Q(T)/2]\gamma_{\max}^2}{1 - \gamma_{\max}^2}\right)\right] \quad (\text{F.12})$$

$$\geq \frac{c'_2}{8}\mu^{1/\alpha-1}\mathbb{E}\left[\mathbb{1}\{N \geq \mu T/2\}\mathbb{1}\{Q(T) \leq \gamma_{\max}^{-2}\}(\mu T/2 - \gamma_{\max}^{-2})c_3\right] \quad (\text{F.13})$$

$$\geq \frac{c'_3}{32}\mu^{1/\alpha-1}(\mu T/2 - \gamma_{\max}^{-2})\Pr(Q(T) \leq \gamma_{\max}^{-2}) \quad (\text{F.14})$$

To bound the probability on RHS, we use the constraint that the regret incurred by  $\pi$  has to be bounded. Notice that if the expression  $\mu^{1/\alpha-1}(\mu T/2 - \gamma_{\max}^{-2})$  is an increasing function of  $T$ , that is if  $\mu^{1/\alpha-1}(\mu T/2 - \gamma_{\max}^{-2}) \sim T^\varepsilon$  for some  $\varepsilon > 0$ , then we would have  $\Pr(Q(T) \leq \gamma_{\max}^{-2}) \sim T^{-\varepsilon}$  since the regret incurred by  $\pi$  is bounded by a constant. More generally, if the allowed regret budget of policy  $\pi$ , denoted by  $R_0(T)$ , is sublinear function of  $\mu^{1/\alpha-1}(\mu T/2 - \gamma_{\max}^{-2})$ , then  $\Pr(Q(T) \leq \gamma_{\max}^{-2}) \lesssim T^{-\varepsilon}$  for some  $\varepsilon > 0$ . This implies that the probability that such a policy queries

less than  $\gamma_{\max}^{-2}$  samples would be small and using Markov's inequality we can conclude that  $\mathbb{E}[Q(T)]$  would be  $\Omega(\gamma_{\max}^{-2})$ . From the assumption on the noise model we have  $\gamma_{\max} \leq c_1 \mu^{\frac{1}{\alpha}-1}$ . Therefore, to obtain the tightest lower bound on expected label complexity, we find the smallest value of  $\mu$  which ensures that  $\mu^{1/\alpha-1}(\mu T/2 - c_1 \mu^{2-\frac{2}{\alpha}})$  is an increasing function of  $T$ . On solving the equation, we get  $\mu \sim T^{-\frac{\alpha}{2-\alpha}}$  and consequently,  $\mathbb{E}[Q(T)] \geq \Omega(T^{\frac{2-2\alpha}{2-\alpha}})$  as required. To evaluate the lower bound for different regret budget  $R_0(T)$ , we just need to evaluate the smallest  $\mu$  that would allow  $R_0(T)$  to be sublinear in  $\mu^{1/\alpha-1}(\mu T/2 - c_1 \mu^{2-\frac{2}{\alpha}})$  and then the corresponding  $\gamma_{\max}$  to obtain the bound.

It is not difficult to see that the assumption of  $\mathcal{H}$  being the class of threshold classifiers was not crucial to the fundamental idea of the proof and it was primarily taken to simplify the description of the hypothesis pair  $h_1$  and  $h_2$ . The proof can be extended to general hypothesis classes by considering  $h_1$  and  $h_2$  such that  $\Pr(\mathcal{D}(h_1, h^*)) = \mu$ ,  $\Pr(\mathcal{D}(h_2, h^*)) = c\mu$  for  $c < 1$  and given  $\mu > 0$  with  $\mathcal{D}(h_2, h^*) \subseteq \mathcal{D}(h_1, h^*)$ . Additionally,  $h_1$  and  $h_2$  also satisfy  $\Pr(\{x : h^*(x) = v \wedge h_1(x) = -v\}) = c_1\mu$  and  $\Pr(\{x : h^*(x) = v \wedge h_2(x) = -v\}) = c_2\mu$  for  $c_1 > c_2 > 0$  and some  $v \in \{-1, 1\}$ . All the arguments in the proof follow exactly for a pair of hypotheses  $h_1$  and  $h_2$  that are chosen in aforementioned manner and consequently, the lower bound  $\mathbb{E}[Q(T)] \geq \Omega(T^{\frac{2-2\alpha}{2-\alpha}})$  holds for general hypothesis classes.

For the case of Massart noise, the proof follows a similar idea. From the condition on the Massart Noise, we have,  $|\eta(x) - 1/2| \geq \gamma_0/2$  for all  $x \in \mathcal{X}$ . Since the slope, i.e.  $|\eta(x) - 1/2|$ , "jumps" at the boundary, we do not need to consider  $h_2$  at all. We just consider  $h_1$  and  $h^*$ . Let  $h_1$  be such that  $\mathcal{D}(h_1, h^*) = \mu$  for some  $\mu > 0$  and  $\gamma_{\max} < 1$  where  $\gamma_{\max} = 2 \max_{x \in \mathcal{D}(h_1, h^*)} |\eta(x) - 1/2|$ , both of which are

independent of  $T$ . Therefore, we can rewrite Eqn. (F.10) as,

$$\begin{aligned}\mathbb{E}[R_\pi(T)] &\geq 2 \sum_{t=1}^T \mathbb{E} \left[ \left| \eta(X_t) - \frac{1}{2} \right| \Pr(q_t = 0 | I_{t-1}) \mathbb{1}\{X_t \in \mathcal{D}(h_1, h^*)\} \exp \left( \frac{-2[m_t/2]\gamma_{\max}^2}{1 - \gamma_{\max}^2} \right) \right] \\ &\geq \gamma_0 \sum_{t=1}^T \mathbb{E} \left[ \Pr(q_t = 0 | I_{t-1}) \mathbb{1}\{X_t \in \mathcal{D}(h_1, h^*)\} \exp \left( \frac{-2[m_t/2]\gamma_{\max}^2}{1 - \gamma_{\max}^2} \right) \right].\end{aligned}\quad (\text{F.15})$$

Again, letting  $N$  denote the number of instances observed in  $\mathcal{D}(h_1, h^*)$  and  $Q(T)$  being the query complexity, we can rewrite the above equation as

$$\mathbb{E}[R_\pi(T)] \geq \gamma_0 \sum_{t=1}^T \mathbb{E} \left[ (N - Q(T)) \exp \left( \frac{-2[Q(T)/2]\gamma_{\max}^2}{1 - \gamma_{\max}^2} \right) \right], \quad (\text{F.16})$$

where we use the trivial inequality  $m_t \leq Q(T)$ . Considering the event  $N \geq \mu T$ , we can write the above equation as

$$\mathbb{E}[R_\pi(T)] \geq \frac{\gamma_0}{4} \frac{-2\gamma_{\max}^2}{1 - \gamma_{\max}^2} \mathbb{E} \left[ \mu T - Q(T) \exp \left( \frac{-Q(T)\gamma_{\max}^2}{1 - \gamma_{\max}^2} \right) \right]. \quad (\text{F.17})$$

Using Jensen inequality and noting that  $xe^{-ax} \leq (ae)^{-1}$  for all  $x \geq 0$ , we can rearrange the above equation as

$$\frac{\mathbb{E}[R_\pi(T)]}{c_\gamma} + \frac{1 - \gamma_{\max}^2}{e\gamma_{\max}^2} \geq T \exp \left( \frac{-\mathbb{E}[Q(T)]\gamma_{\max}^2}{1 - \gamma_{\max}^2} \right), \quad (\text{F.18})$$

where  $c_\gamma = \frac{\gamma_0}{4} \frac{-2\gamma_{\max}^2}{1 - \gamma_{\max}^2}$ . Noting that  $\mathbb{E}[R_\pi(T)]$  is a sublinear function of  $T$ , i.e.,  $\mathbb{E}[R_\pi(T)]$  is  $O(T^\varepsilon)$  for some  $\varepsilon \in (0, 1)$ , we can rearrange the above equation to obtain  $\mathbb{E}[Q(T)]$  is  $\Omega(\log T)$ , as required.

## F.5 Proof of Theorem 7.5.1

We begin with stating a result from [309] that establishes the relationship between the empirical error and true error rate of any hypothesis  $h$ .

**Lemma F.5.1.** Let  $\mathcal{Z}$  be a set of  $M$  i.i.d.  $(X, Y)$ -samples under distribution  $\mathbb{P}$ . For all  $h \in \mathcal{H}$ , we have, with probability at least  $1 - \delta$ ,

$$|\epsilon_{\mathbb{P}}(h) - \epsilon_{\mathcal{Z}}(h)| \leq \Delta(M, \delta)$$

We define

$$d(\mathcal{H}_k) := \min_{h \notin \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h) - \min_{h \in \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h),$$

which captures the hardness of the hypothesis testing problem.

To analyse the regret of RW-OLA, we define the following random process that captures the progress of the algorithm towards  $h^*$ .

**Definition F.5.2.** Consider a random process  $\{W_k\}_{k \geq 1}$ , indexed by the epoch index  $k$  and initialized with  $W_1 = 0$ . The evolution of this random process from  $W_k$  to  $W_{k+1}$  for any  $k \geq 1$  is determined as follows:

1. If  $d(\mathcal{H}_k) \geq 4\Delta(M, 1 - \sqrt{p})$ , then  $W_{k+1} = W_k + 1$  if the verification passes with  $d(\mathcal{H}_{k+1}) \geq 4\Delta(M, 1 - \sqrt{p})$ . However, if the verification passes with  $d(\mathcal{H}_{k+1}) < 4\Delta(M, 1 - \sqrt{p})$  or fails altogether, then  $W_{k+1} = W_k - 1$ .
2. If  $0 \leq d(\mathcal{H}_k) < 4\Delta(M, 1 - \sqrt{p})$ , then  $W_{k+1} = W_k + 1$  if the verification passes with  $d(\mathcal{H}_{k+1}) \geq 4\Delta(M, 1 - \sqrt{p})$  or fails altogether. Otherwise,  $W_{k+1} = W_k - 1$ .
3. When  $d(\mathcal{H}_k) < 0$ ,  $W_{k+1} = W_k + 1$  when the verification passes and  $W_{k+1} = W_k - 1$  when the verification fails.

The proof consists of three main lemmas. The first lemma establishes that the random process  $\{W_k\}_{k \geq 1}$  is biased towards positive increments, that is, with each epoch, the value of  $W_k$  is more likely to increase than it is to decrease. The second lemma relates the progress of algorithm with the value of  $W_k$ . In

particular, it shows that as the probability mass of the region of disagreement shrinks exponentially with the value of  $W_k$ . The third lemma shows that the bias of the random process ensures that the probability mass of the region of disagreement shrinks exponentially also with the epoch index, as in the case of OLA (See Appendix F.3). The final bound on label complexity then follows the same steps as in the proof for the label complexity of OLA.

The following lemma shows that the value of the random process  $W_k$  increases in each epoch with probability  $p > 1/2$ .

**Lemma F.5.3.** *For all epoch  $k$ ,  $\Pr(W_{k+1} = W_k + 1) \geq p$ .*

*Proof.* Based on how  $W_{k+1}$  is defined, we consider following three cases:

1.  $d(\mathcal{H}_k) \geq 4\Delta(M, 1 - \sqrt{p})$ : From Lemma F.5.1 we can conclude that  $|\epsilon_P(h) - \epsilon_Z(h)| \leq \Delta(M, 1 - \sqrt{p})$  holds for all  $h \in \mathcal{H}$  with probability at least  $\sqrt{p}$ . Specifically, we have,

$$\begin{aligned} \min_{h \notin \mathcal{H}_k} \epsilon_{Z'_k}(h) &\geq \min_{h \notin \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h) - \Delta(M, 1 - \sqrt{p}), \\ \min_{h \in \mathcal{H}_k} \epsilon_{Z'_k}(h) &\leq \min_{h \in \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h) + \Delta(M, 1 - \sqrt{p}). \end{aligned}$$

Upon adding the two equations, we obtain,

$$\begin{aligned} \min_{h \notin \mathcal{H}_k} \epsilon_{Z'_k}(h) - \min_{h \in \mathcal{H}_k} \epsilon_{Z'_k} &\geq \min_{h \notin \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h) - \min_{h \in \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h) - 2\Delta(M, 1 - \sqrt{p}) \\ &\geq d(\mathcal{H}_k) - 2\Delta(M, 1 - \sqrt{p}) \\ &\geq 4\Delta(M, 1 - \sqrt{p}) - 2\Delta(M, 1 - \sqrt{p}) \\ &\geq 2\Delta(M, 1 - \sqrt{p}). \end{aligned}$$

The second line follows from the definition of  $d(\mathcal{H}_k)$ . Consequently, we can conclude that the algorithm zooms in with probability at least  $\sqrt{p}$ . On

applying Lemma F.5.1 over samples in  $\mathcal{H}_{k+1}$ , we obtain that the following relation holds with probability at least  $\sqrt{p}$ :

$$\begin{aligned}
\min_{h \notin \mathcal{H}_{k+1}} \epsilon_{\mathbb{P}_{\mathcal{D}_k}}(h) &\geq \min_{h \notin \mathcal{H}_{k+1}} \epsilon_{\mathcal{Z}_k}(h) - \Delta(M, 1 - \sqrt{p}) \\
&\geq \epsilon_{\mathcal{Z}_k}(h_k^*) + 6\Delta(M, 1 - \sqrt{p}) - \Delta(M, 1 - \sqrt{p}) \\
&\geq \epsilon_{\mathbb{P}_{\mathcal{D}_k}}(h_k^*) - \Delta(M, 1 - \sqrt{p}) + 5\Delta(M, 1 - \sqrt{p}) \\
&\geq \epsilon_{\mathbb{P}_{\mathcal{D}_k}}(h^*) + 4\Delta(M, 1 - \sqrt{p}).
\end{aligned}$$

This implies that  $d(\mathcal{H}_{k+1}) \geq 4\Delta(M, 1 - \sqrt{p})$  holds with probability at least  $\sqrt{p}$ . Therefore,  $\Pr(W_{k+1} = W_k + 1 | d(\mathcal{H}_k) \geq 4\Delta(M, 1 - \sqrt{p})) \geq (\sqrt{p})^2 = p$ .

2.  $0 \leq d(\mathcal{H}_k) < 4\Delta(M, 1 - \sqrt{p})$ :

Since  $W_{k+1} = W_k + 1$  when the algorithm zooms out, it suffices to show that when the algorithm zooms in,  $d(\mathcal{H}_{k+1}) \geq 4\Delta(M, 1 - \sqrt{p})$  with probability at least  $\sqrt{p}$ . By applying Lemma F.5.1, we obtain that the following relation holds with probability at least  $\sqrt{p}$ :

$$\begin{aligned}
\min_{h \notin \mathcal{H}_{k+1}} \epsilon_{\mathbb{P}_{\mathcal{D}_k}}(h) &\geq \min_{h \notin \mathcal{H}_{k+1}} \epsilon_{\mathcal{Z}_k}(h) - \Delta(M, 1 - \sqrt{p}) \\
&\geq \epsilon_{\mathcal{Z}_k}(h_k^*) + 6\Delta(M, 1 - \sqrt{p}) - \Delta(M, 1 - \sqrt{p}) \\
&\geq \epsilon_{\mathbb{P}_{\mathcal{D}_k}}(h_k^*) - \Delta(M, 1 - \sqrt{p}) + 5\Delta(M, 1 - \sqrt{p}) \\
&\geq \epsilon_{\mathbb{P}_{\mathcal{D}_k}}(h^*) + 4\Delta(M, 1 - \sqrt{p})
\end{aligned}$$

Therefore,  $\Pr(W_{k+1} = W_k + 1 | 0 \leq d(\mathcal{H}_k) < 4\Delta(M, 1 - \sqrt{p})) \geq \sqrt{p} \geq p$ .

3.  $d(\mathcal{H}_k) < 0$ : It suffices to show that the probability that the algorithm zooms out is at least  $\sqrt{p}$ . Since  $d(\mathcal{H}_k) < 0$ , we know that  $\min_{h \in \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h) < \min_{h \in \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_k}}(h)$ . Once again, from Lemma F.5.1 we can conclude that the

following two relations hold with probability at least  $\sqrt{p}$ :

$$\min_{h \notin \mathcal{H}_k} \epsilon_{\mathcal{Z}'_k}(h) \leq \min_{h \notin \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h) + \Delta(M, 1 - p)$$

$$\min_{h \in \mathcal{H}_k} \epsilon_{\mathcal{Z}'_k}(h) \geq \min_{h \in \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h) - \Delta(M, 1 - p).$$

On adding the above equations and noting that  $\min_{h \notin \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h) < \min_{h \in \mathcal{H}_k} \epsilon_{\mathbb{P}_{\mathcal{D}_{r(k)}}}(h)$ , we have

$$\min_{h \notin \mathcal{H}_k} \epsilon_{\mathcal{Z}'_k}(h) - \min_{h \in \mathcal{H}_k} \epsilon_{\mathcal{Z}'_k}(h) \leq 2\Delta(M, 1 - \sqrt{p}).$$

Consequently, the algorithm zooms out with probability at least  $\sqrt{p}$  implying that  $\Pr(W_{k+1} = W_k + 1 | 0 \leq d(\mathcal{H}_k) < 0) \geq \sqrt{p} \geq p$ .

On combining the results from all the cases, we have  $\Pr(W_{k+1} = W_k + 1) \geq p$ , as desired.  $\square$

Since either  $W_{k+1} = W_k + 1$  or  $W_{k+1} = W_k - 1$ ,  $W_k$  is a random walk process with positive bias at least  $p$ . The next lemma establishes the exponential convergence of RoD with  $W_k$ .

**Lemma F.5.4.** *There exists  $c_2 < 1$  such that  $\mathbb{E}[\phi(\mathcal{D}_{r(k)}) | W_k = w] \leq c_2^w$ .*

*Proof.* We first show that  $\mathbb{E}[\phi(\mathcal{D}_{k+1}) | \phi(\mathcal{D}_k), r(k+1) = k, h^* \in \mathcal{H}_k] \leq c\phi(\mathcal{D}_k)$ . Let  $c = 32\theta c_0 m^{-\frac{1}{2}} < 1$  and  $\mathcal{H}_k^\theta = \left\{ h \in \mathcal{H}_k, \rho(h, h^*) > \frac{c\phi(\mathcal{D}_k)}{\theta} \right\}$ . Furthermore, using the definition of  $\Delta(M, 1 - \sqrt{p})$ , one can conclude that  $\Delta(M, 1 - \sqrt{p}) \leq 4/\sqrt{m}$ . Note that

if  $h \in \mathcal{H}_k^\theta$ , then

$$\begin{aligned}
\epsilon_{\mathbb{P}|\mathcal{D}_k}(h) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h^*) &= \frac{\epsilon_{\mathbb{P}}(h) - \epsilon_{\mathbb{P}}(h^*)}{\phi(\mathcal{D}_k)} \\
&\geq \frac{\rho(h, h^*)}{c_0 \phi(\mathcal{D}_k)} \\
&\geq \frac{32\phi(\mathcal{D}_k)}{\sqrt{m}} \frac{1}{\phi(\mathcal{D}_k)} \\
&\geq \frac{32}{\sqrt{m}} \\
&\geq 8\Delta(M, 1 - \sqrt{p}).
\end{aligned}$$

Thus by Lemma F.5.1, we can conclude that the following relation holds with probability  $p$ :

$$\begin{aligned}
\epsilon_{\mathcal{Z}_k}(h) - \epsilon_{\mathcal{Z}_k}(h_k^*) &\geq \epsilon_{\mathcal{Z}_k}(h) - \epsilon_{\mathcal{Z}_k}(h^*) \\
&\geq \epsilon_{\mathbb{P}|\mathcal{D}_k}(h) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h^*) - 2\Delta(M, 1 - \sqrt{p}) \\
&> 8\Delta(M, 1 - \sqrt{p}) - 2\Delta(M, 1 - \sqrt{p}) \\
&> 6\Delta(M, 1 - \sqrt{p})
\end{aligned}$$

This implies that for all  $h \in \mathcal{H}_k^\theta$ ,  $h \notin \mathcal{H}_{k+1}$  whenever the algorithm zooms in. Using the definition of  $\theta$ , we obtain  $\phi(\mathcal{D}_{k+1}) \leq \phi(\Psi(\mathcal{H}_k \setminus \mathcal{H}_k^\theta)) \leq \frac{c\phi(\mathcal{D}_k)}{\theta} \cdot \theta = c\phi(\mathcal{D}_k)$ . Therefore,  $\mathbb{E}[\phi(\mathcal{D}_{k+1})|\phi(\mathcal{D}_k), r(k+1) = k] \leq c_2\phi(\mathcal{D}_k)$  as desired, where  $c_2 = c_0 p + 1 - c_0 < 1$ .

Note that for the general case, for each  $k$ , we can find a sequence of  $(i_1, i_2, \dots, i_{j_k}, k)$  where  $r(i_{l+1}) = i_l$  for  $l = 1, 2, \dots, i_{j_k}$  and  $r(k) = i_{j_k}$ . Let  $j^*(k)$  be the largest integer such that  $h^* \in \mathcal{H}_{i_{j^*(k)}}^*$ . Then, by the definition of  $W_k$ , we have  $j^*(k) \geq W_k$ . Note that  $\mathcal{H}_{i_{l+1}} \subseteq \mathcal{H}_{i_l}$  for all  $l = 1, 2, \dots, i_{j_k}$ , we can apply the inequality  $\mathbb{E}[\phi(\mathcal{D}_{k+1})|\phi(\mathcal{D}_k), r(k+1) = k, h^* \in \mathcal{H}_k] \leq c_2\phi(\mathcal{D}_k)$   $j^*(k)$  times, which gives us

$$\mathbb{E}[\phi(\mathcal{D}_{r(k)})|W_k = w] \leq \mathbb{E}[\phi(\mathcal{D}_{i_{j^*(k)}})|W_k = w] \leq c_2^{j^*(k)} \leq c_2^w$$

as desired.  $\square$

The following lemma shows that this exponential convergence can be extended from the random process to that over epoch index.

**Lemma F.5.5.** *There exists  $c_3 < 1$  such that  $\mathbb{E}[\phi(\mathcal{D}_{r(k)})] \leq c_4 \cdot c_3^k$ .*

*Proof.* By tower property and Lemma F.5.4, we have  $\mathbb{E}[\phi(\mathcal{D}_{r(k)})] = \mathbb{E}[\mathbb{E}[\phi(\mathcal{D}_{r(k)})|W_k]] = \mathbb{E}[c_2^{W_k}]$ . Recall that  $W_k$  is a random walk process over integers with a bias of at least  $p$ . Let  $V_k = \text{Bin}(k, p)$  be a binomial random variable. Then for any  $n \geq 0$  we have  $P(W_k \geq n) \geq P(2V_k - k \geq n)$ . Therefore, by interchanging the sum order and using the Moment Generating Function for Binomial random variable, we have

$$\mathbb{E}[c_2^{W_k}] \leq \mathbb{E}[c_2^{2V_k}] = \left( \frac{1-p+c_2^2p}{c_2} \right)^k = c_3^k, \quad (\text{F.19})$$

$$\text{where } c_3 = \frac{1-p+c_2^2p}{c_2} < 1. \quad \square$$

The rest of the proof is similar to that of Theorem 7.4.4. Let  $Q(t)$  denote the label complexity at time  $t$ . Define  $S(t) = \left( \frac{1}{c_3} \right)^{\frac{Q(t)}{2M}} - \left( \frac{1}{c_3} \right) \left[ \left( \frac{1}{c_3} \right)^{\frac{1}{2M}} - 1 \right] t$ . Using the same arguments as in Appendix F.3, we can conclude that  $S(t)$  is supermartingale. Consequently, optional stopping theorem implies  $\mathbb{E}[S(T)] \leq \mathbb{E}[S(0)] = 1$ . Plugging in the value of  $S(T)$  along with Jensen's inequality yields

$$\mathbb{E}[Q(T)] \leq \frac{2md}{\log \frac{1}{c_3}} \log(2T+1) = O(d \log T),$$

as required.

Next we show that  $S(t)$  is a supermartingale. Since  $Q(t) \leq 2(k_t + 1)M$ , we have

$$\begin{aligned} \Pr(Q(t+1) = Q(t) + 1 | S(1), S(2), \dots, S(t)) &= \Pr(q_{t+1} = 1 | S(1), S(2), \dots, S(t)) \\ &= \mathbb{E}[\phi(\mathcal{D}_{r(k_t)}) | Q(t)] \\ &\leq c_3^{\frac{Q(t)}{2M} - 1}. \end{aligned} \quad (\text{F.20})$$

Therefore,

$$\begin{aligned} &\mathbb{E}[S_{t+1} | S(1), S(2), \dots, S(t)] \\ &= \mathbb{E}\left[\left(\frac{1}{c_3}\right)^{\frac{Q(t)}{2M}} - \left(\frac{1}{c_3}\right)\left(\left(\frac{1}{c_3}\right)^{\frac{1}{2M}} - 1\right)t | S(1), S(2), \dots, S(t)\right] \\ &\leq \left(\frac{1}{c_3}\right)^{\frac{Q(t)+1}{2M}} \left(\frac{1+c}{2}\right)^{\frac{Q(t)}{2M}-1} + \left(\frac{1}{c_3}\right)^{\frac{Q(t)}{2M}} \left(1 - \left(\frac{1+c}{2}\right)^{\frac{Q(t)-1}{2M}}\right) \\ &\quad - \left(\frac{1}{c_3}\right)\left[\left(\frac{1}{c_3}\right)^{\frac{1}{2M}} - 1\right](t+1) \\ &= \left(\frac{1}{c_3}\right)^{\frac{Q(t)}{2M}} - \left(\frac{1}{c_3}\right)\left[\left(\frac{1}{c_3}\right)^{\frac{1}{2M}} - 1\right]t = S(t) \end{aligned} \quad (\text{F.21})$$

as desired. Then by optional stopping theorem,

$$\begin{aligned} &\mathbb{E}\left[\left(\frac{1}{c_3}\right)^{\frac{Q(T)}{2M}} - \left(\frac{1}{c_3}\right)\left(\left(\frac{1}{c_3}\right)^{\frac{1}{2M}} - 1\right)t\right] \\ &= \mathbb{E}[S(T)] \leq \mathbb{E}[S(0)] = 1. \end{aligned} \quad (\text{F.22})$$

Hence,

$$\mathbb{E}\left[\left(\frac{1}{c_3}\right)^{\frac{Q(T)}{2M}}\right] \leq \left(\frac{1}{c_3}\right)\left(\left(\frac{1}{c_3}\right)^{\frac{1}{2M}} - 1\right)T + 1. \quad (\text{F.23})$$

Since  $f(x) = \log x$  is concave, by Jensen's Inequality we have,

$$\mathbb{E}[Q(T)] \leq M \log\left(\frac{1}{c_3}\right) \left[\left(\frac{1}{c_3}\right)\left(\left(\frac{1}{c_3}\right)^{\frac{1}{2M}} - 1\right)T + 1\right]. \quad (\text{F.24})$$

Since  $M \leq md$ ,  $c_3 < \frac{1}{2}$  and  $(\frac{1}{c_3})((\frac{1}{c_3})^{\frac{1}{2M}} - 1) < 2$ , we have

$$\mathbb{E}[Q(T)] \leq \frac{2md}{\log \frac{1}{c_3}} \log(2T + 1) = O(d \log T).$$

as desired.

## F.6 Proof of Theorem 7.5.2

We first state and prove an auxiliary lemma that will be used later in the proof.

**Lemma F.6.1.** *There exists  $c_5 > 0$  such that  $\mathbb{E}[\phi(\mathcal{D}_{k+1})|\phi(\mathcal{D}_k)] \geq c_5\phi(\mathcal{D}_k)$ .*

*Proof.* Let  $c' = 8\theta'c_0m^{-\frac{1}{2}}$  and  $\mathcal{V}_k^{\theta'} = \left\{h \in \mathcal{H}_k, \rho(h, h^*) \leq \frac{c'\phi(\mathcal{D}_k)}{\theta'}\right\}$ . From the definition of  $\Delta(M, 1 - \sqrt{p})$ , one can note that  $\Delta(M, 1 - \sqrt{p}) \geq 2/\sqrt{m}$ . Assume that there exists a constant  $c'_0$  such that  $c'_0(d(h, h^*)) \leq \rho(h, h^*)$  holds for all  $h \in \mathcal{H}$ . For any  $h \in \mathcal{V}_k^{\theta'}$ , we have

$$\begin{aligned}\epsilon_{\mathbb{P}|\mathcal{D}_k}(h) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h^*) &= \frac{\epsilon_{\mathbb{P}}(h) - \epsilon_{\mathbb{P}}(h^*)}{\phi(\mathcal{D}_k)} \\ &\leq \frac{\rho(h, h^*)}{c'_0\phi(\mathcal{D}_k)} \\ &\leq \frac{8\phi(\mathcal{D}_k)}{\sqrt{m}} \frac{1}{\phi(\mathcal{D}_k)} \\ &\leq \frac{8}{\sqrt{m}} \\ &\leq 4\Delta(M, 1 - \sqrt{p}).\end{aligned}$$

Thus, by Lemma F.5.1, we can conclude that the following holds with probability  $p$ .

$$\begin{aligned}\epsilon_{\mathcal{Z}_k}(h) - \epsilon_{\mathcal{Z}_k}(h_k^*) &\leq \epsilon_{\mathcal{Z}_k}(h) - \epsilon_{\mathcal{Z}_k}(h^*) \\ &\leq \epsilon_{\mathbb{P}|\mathcal{D}_k}(h) - \epsilon_{\mathbb{P}|\mathcal{D}_k}(h^*) + 2\Delta(M, 1 - \sqrt{p}) \\ &\leq 4\Delta(M, 1 - \sqrt{p}) + 2\Delta(M, 1 - \sqrt{p}) \\ &\leq 6\Delta(M, 1 - \sqrt{p})\end{aligned}$$

This indicates that for all  $h \in \mathcal{V}_k^{\theta'}$ ,  $h \in \mathcal{H}_{k+1}$  whenever the algorithm zooms in. By the definition of  $\theta$ , we have  $\phi(\mathcal{D}_{k+1}) \geq \phi(\Psi(\mathcal{V}_k^{\theta'})) \geq \frac{c'\phi(\mathcal{D}_k)}{\theta'} \cdot \theta' = c'\phi(\mathcal{D}_k)$  with probability at least  $p$ . Therefore  $\mathbb{E}[\phi(\mathcal{D}_{k+1})|\phi(\mathcal{D}_k)] \geq c_5\phi(\mathcal{D}_k)$  as desired, where  $c_5 = c'p$ .  $\square$

To bound the regret of RW-OLA, using a technique similar to the one used to bound the label complexity. We first define a random process  $\{U_k\}_{k \geq 1}$ , where  $k$  denotes the epoch index. The random process is initialized with  $U_1 = 0$ .

**Definition F.6.2.** The random process  $\{U_k\}_{k \geq 1}$  evolves as follows.

$$U_{k+1} = \begin{cases} U_k & \text{if } d(\mathcal{H}_k) \geq 0, \\ U_k + 1 & \text{if } d(\mathcal{H}_k) < 0 \text{ and verification passes,} \\ U_k - 1 & \text{if } d(\mathcal{H}_k) < 0 \text{ and verification fails.} \end{cases}$$

For the above defined process, note that using same argument as used in the proof of Lemma F.5.3 for the case of  $d(\mathcal{H}_k) < 0$ , we can conclude that  $\Pr(U_{k+1} = U_k + 1) \leq 1 - p$ . The following lemma, which is a counterpart of Lemma F.5.4, characterizes the evolution of regret as a function of  $U_k$ .

**Lemma F.6.3.** Let  $R_k$  be the regret at epoch  $k$ . Then, there exists  $c_6 > 0$  such that  $\mathbb{E}[R_k|U_k] \leq c_6^{U_k} \cdot 2M$ .

*Proof.* Firstly, note that whenever  $h^* \in \mathcal{H}_k$ , RW-OLA incurs no regret during epoch  $k$ . When  $h^* \notin \mathcal{H}_k$ , and let  $r' < k$  be the last time such that  $h^* \in \mathcal{H}_{r'}$ , then, for epoch  $k$  we have,

$$\Pr(r_t = 1) \leq \Pr(X_t \notin \mathcal{D}_k) = \phi(\mathcal{D}_{r'}) - \phi(\mathcal{D}_k).$$

From the definition of  $\{U_l\}_{l \geq 1}$  (See Definition F.6.2), we note that  $U_k$  denotes the number of times the algorithm has zoomed in from epoch  $r'$  to epoch  $k$ . Using the result of Lemma F.6.1, we can conclude that

$$\Pr(r_t = 1|U_k) \leq \phi(\mathcal{D}_{r'}) - \phi(\mathcal{D}_k) \leq \frac{1 - c_5^{U_k}}{c_5^{U_k}} \phi(\mathcal{D}_{r'}) \leq (1/c_5)^{U_k} \phi(\mathcal{D}_k).$$

Let  $N$  denote the random number of points seen by the algorithm during epoch  $k$  and let the corresponding time instants be given as  $\{t', t' + 1, t' + 2, \dots, t' + N - 1\}$ . Consequently, we have,

$$\begin{aligned}\mathbb{E}[R_k|U_k] &= \mathbb{E}\left[\sum_{t=t'}^{t'+N-1} \mathbb{1}_{\{r_t = 1\}}|U_k\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[\sum_{t=t'}^{t'+N-1} \mathbb{1}_{\{r_t = 1\}}|U_k, N\right]|U_k\right] \\ &= \mathbb{E}[N \Pr(r_t = 1|U_k)|U_k] \\ &\leq (1/c_5)^{U_k} \phi(\mathcal{D}_k) \mathbb{E}[N|U_k].\end{aligned}$$

Since the probability mass of the disagreement region is  $\phi(\mathcal{D}_k)$ , the expected number of points seen until a query is  $1/\phi(\mathcal{D}_k)$ . Moreover, since the epoch terminates once the algorithm has queried  $2M$  points,  $\mathbb{E}[N|U_k] = 2M/\phi(\mathcal{D}_k)$ . Plugging this back into the expression for  $\mathbb{E}[R_k|U_k]$ , we obtain,

$$\mathbb{E}[R_k|U_k] = (1/c_5)^{U_k} \cdot 2M,$$

as desired for  $c_6 = 1/c_5$ . □

**Lemma F.6.4.** *If  $p > \frac{1}{2} - \frac{1}{4c} + \sqrt{\frac{1-c}{4c}}$  where  $c = 32\theta c_0 m^{-\frac{1}{2}}$ , there exists  $c_7 > 0$  such that*

$$\mathbb{E}[R_k] \leq c_7 M.$$

*Proof.* By tower property and using arguments similar to the ones used in the proof of Lemma F.5.5, we have,

$$\mathbb{E}[R_k] = \mathbb{E}[\mathbb{E}[R_k|U_k]] = \mathbb{E}[c_6^{U_k}]. \quad (\text{F.25})$$

Note that  $U_k$  is a random walk process with negative bias at least  $p$ . By interchanging the sum order and using the Moment Generating Function for Binomial

mial random variable, we have

$$\mathbb{E}[c_6^{U_k}] \leq \left(\frac{1-p^2}{p^2}\right)^k, \quad (\text{F.26})$$

as desired.  $\square$

Using a supermartingale based argument similar to one used in proof of Theorem 7.5.1, we can conclude that

$$\mathbb{E}[R(T)] \leq c_7 m d \log T,$$

as desired.

APPENDIX G  
CHAPTER 8

## G.1 Proof of Lemma 8.4.2

We begin proving the claim on the error probability. We denote the true underlying signal (the number of defectives) with  $x^* \in \mathcal{X} = \{0, 1, 2, \dots, d\}$  and the observed signal with  $y \in \mathcal{Y} = \{0, 1, \dots, K-1\}$ . Recall that  $Q_r(x)$  denotes the result of a noiseless test with threshold  $r$  on a group with  $x$  defectives. For simplicity, let  $y^* = Q_r(x^*)$ . Lastly, let  $p = (p_0, p_1, p_2, \dots, p_{K-1})$  denote the distribution  $P(\cdot|x^*)$ . From our assumption on the noise model, we have that  $p_{Q_r(x^*)} > p_y + \rho$  for all  $y \in \mathcal{Y} \setminus \{y^*\}$ .

For any  $y \in \{0, 1, \dots, K-1\}$  and  $y \neq y^*$ , define the random variable  $W_t(y) = \mathbb{1}\{Y_t = y\} - \mathbb{1}\{Y_t = y^*\}$ . Consequently,  $\mathbb{E}[Y_t(y)] = p_y - p_{y^*} < -\rho < 0$ . Clearly,  $W_t(y)$  is a sequence of i.i.d. random variables. Since,  $W_t(y) \in [-1, 1]$ , it is also sub-Gaussian with  $\sigma^2 = 1$ . Noting that  $\tau_{\text{sq-test}}$  is the random number of samples taken in the

test, we can write the probability of error as,

$$\begin{aligned}
\Pr(\text{err}) &= \Pr\left(\hat{i}_{\tau_{\text{sq-test}}} \neq y^*\right) \\
&\leq \max_{y \neq y^*} \Pr\left(\hat{i}_{\tau_{\text{sq-test}}} = y\right) \\
&\leq \max_{y \neq y^*} \Pr\left(\hat{p}_y(t) - \hat{p}_{\hat{j}_t}(t) > \sqrt{\frac{5}{t} \log\left(\frac{6 \log t}{\sqrt{q}}\right)}, t = \tau_{\text{sq-test}}\right) \\
&\leq \max_{y \neq y^*} \Pr\left(\hat{p}_y(t) - \hat{p}_{y^*}(t) > \sqrt{\frac{5}{t} \log\left(\frac{6 \log t}{\sqrt{q}}\right)}, t = \tau_{\text{sq-test}}\right) \\
&\leq \max_{y \neq y^*} \Pr\left(\bigcup_t \left\{ \frac{1}{t} \sum_{s=1}^t W_s(y) > \sqrt{\frac{5}{s} \log\left(\frac{6 \log s}{\sqrt{q}}\right)} \right\}\right) \\
&\leq q
\end{aligned}$$

The last step follows from the relation on the sub-Gaussian random variables described in []. Hence, the probability of error is bounded by  $q$  as required.

We now prove the upper bound on the number of samples taken in the test. Let  $s_0 = \frac{20}{\rho^2} \log\left(\frac{2}{\lambda} \log\left(\frac{40}{\lambda\rho^2}\right)\right)$  and  $s_1 = \lceil s_0 \rceil$  where  $\lambda = \frac{\sqrt{q}}{6}$ . For this choice of  $s_0$ , one can note that  $\vartheta(s) \leq \rho/2$  for all  $s \geq s_1$ , where  $\vartheta(s) = \sqrt{\frac{5}{s} \log\left(\frac{6 \log s}{\sqrt{q}}\right)}$ , the threshold used in the sequential test. For  $n \geq s_1$ , we have,

$$\begin{aligned}
\Pr(\tau_{\text{sq-test}} > n) &= \Pr(\tau_{\text{sq-test}} > n | \hat{i}_n = y^*) \Pr(\hat{i}_n = y^*) + \Pr(\tau_{\text{sq-test}} > n | \hat{i}_n \neq y^*) \Pr(\hat{i}_n \neq y^*) \\
&\leq \Pr(\tau_{\text{sq-test}} > n | \hat{i}_n = y^*) + \Pr(\hat{i}_n \neq y^*) \\
&\leq \left[ \sum_{y \in \mathcal{Y} \setminus \{y^*\}} \Pr(\tau > n | \hat{i}_n = y^*, \hat{j}_n = y) \Pr(\hat{j}_n = y | \hat{i}_n = y^*) \right] + \Pr(\hat{i}_n \neq y^*).
\end{aligned}$$

We bound each of the term separately as follows.

$$\begin{aligned}
\Pr(\hat{i}_n \neq y^*) &\leq \max_{y \in \mathcal{Y} \setminus \{y^*\}} \Pr(\hat{i}_n = y) \\
&\leq \max_{y \in \mathcal{Y} \setminus \{y^*\}} \Pr(\hat{p}_y(n) - \hat{p}_{y^*}(n) > 0) \\
&\leq \max_{y \in \mathcal{Y} \setminus \{y^*\}} \Pr\left(\frac{1}{n} \sum_{s=1}^n W_s(y) > 0\right) \\
&\leq \max_{y \in \mathcal{Y} \setminus \{y^*\}} \Pr\left(\frac{1}{n} \sum_{s=1}^n W_s(y) - \mathbb{E}[W_n(y)] > -\mathbb{E}[W_n(y)]\right) \\
&\leq \max_{y \in \mathcal{Y} \setminus \{y^*\}} \exp\left(-\frac{n}{2}(p_y - p_{y^*})^2\right) \\
&\leq \exp\left(-\frac{n}{2}\rho^2\right).
\end{aligned}$$

Similarly, we consider,

$$\begin{aligned}
\Pr(\tau_{\text{sq-test}} > n | \hat{i}_n = y^*, \hat{j}_n = y) &\leq \Pr\left(\hat{p}_{y^*}(n) - \hat{p}_y(n) < \sqrt{\frac{5}{n} \log\left(\frac{6 \log n}{\sqrt{q}}\right)}\right) \\
&\leq \Pr\left(-\frac{1}{n} \sum_{s=1}^n W_s(y) < \frac{\rho}{2}\right) \\
&\leq \Pr\left(\frac{1}{n} \sum_{s=1}^n W_s(y) - \mathbb{E}[Y_n(j)] > -\frac{\rho}{2} - (p_y - p_{y^*})\right) \\
&\leq \Pr\left(\frac{1}{n} \sum_{s=1}^n W_s(y) - \mathbb{E}[Y_n(j)] > \frac{\rho}{2}\right) \\
&\leq \exp\left(-\frac{n}{8}\rho^2\right).
\end{aligned}$$

On combining the expressions, we have,

$$\begin{aligned}
\Pr(\tau_{\text{sq-test}} > n) &\leq \left[ \sum_{y \in \mathcal{Y} \setminus \{y^*\}} \Pr(\tau > n | \hat{i}_n = y^*, \hat{j}_n = y) \Pr(\hat{j}_n = y | \hat{i}_n = y^*) \right] + \Pr(\hat{i}_n \neq y^*) \\
&\leq \left[ \sum_{j \neq i^*} \exp\left(-\frac{n}{8}\rho^2\right) \Pr(\hat{j}_n = y | \hat{i}_n = y^*) \right] + \exp\left(-\frac{n}{2}\rho^2\right) \\
&\leq \exp\left(-\frac{n}{8}\rho^2\right) + \exp\left(-\frac{n}{2}\rho^2\right) \\
&\leq 2 \exp\left(-\frac{n}{8}\rho^2\right).
\end{aligned}$$

We can write  $\mathbb{E}[\tau_{\text{sq-test}}]$  in terms of the sum of the tail probabilities  $\Pr[\tau_{\text{sq-test}} \geq n]$  as

$$\begin{aligned}
\mathbb{E}[\tau_{\text{sq-test}}] &= \sum_{n=0}^{\infty} \Pr[\tau_{\text{sq-test}} > n] \\
&= s_1 + \sum_{n=s_1+1}^{\infty} \Pr[\tau_{\text{sq-test}} > n] \\
&\leq s_1 + \sum_{n=s_1+1}^{\infty} 2 \exp\left(-\frac{n}{8}\rho^2\right) \\
&\leq s_1 + 2 \int_{s_1+1}^{\infty} \exp\left(-\frac{x}{8}\rho^2\right) dx \\
&\leq s_1 + 2 \left[ \frac{8}{\rho^2} \exp(-s_1\mu^2/(8\sigma^2)) \right] \\
&\leq s_1 + \frac{16}{\rho^2} \\
&\leq 2s_1,
\end{aligned}$$

which gives that  $\mathbb{E}[\tau_{\text{sq-test}}] \leq \frac{40}{\rho^2} \log\left(\frac{12}{\sqrt{q}} \log\left(\frac{240}{\sqrt{q}\rho^2}\right)\right) + 2$  as required.

## G.2 Proof of Lemma 8.4.3

The proof of this lemma is very similar to that of Lemma 3.4.4 and to the analysis in Appendix A.3.2. and. We reproduce the proof here for completeness and to establish some notation that will be used in other proofs.

Consider a population of  $n$  items indexed from 1 to  $n$  and without loss of generality assume that the defective item is item number 1. We construct a binary tree over this population of  $n$  items where each node represents the group to be tested. Each child node is formed by splitting the group corresponding to the parent node into two equal parts. For concreteness, we assign the left child to the group of first  $\lceil n/2 \rceil$  items and the right child corresponds to the group

the remaining  $\lfloor n/2 \rfloor$  items. We begin with at the root node corresponding to the entire population and the above recursive process is continued till a node corresponds to a single item, referred to as the leaf node. The depth of a node is the number of nodes between the root node and the given node in the path connecting the two. Thus, the root node is at depth 0, its children are at a depth 1, its grandchildren at depth 2 and so on.

For the analysis, we divide the tree into a sequence of sub-trees given by  $\mathcal{T}_0, \mathcal{T}_1, \dots$  for  $i = 0, 1, 2, \dots, \lceil \log_2 n \rceil$  which are defined as follows. Consider the nodes on the path joining the root node to the target node. Such a path is unique as the underlying graph is a tree. Let  $v_i$  denote the node on this path that is at a distance of  $i$  from the target node.  $\mathcal{T}_i$  is defined to be tree that contains the node  $v_i$  along with the tree rooted at the child that does not contain the target. For example, if the defective item is the item with index 1, then the sub-tree  $\mathcal{T}_i$  contains the left most node at depth  $\lceil \log_2 n \rceil - i$  along with its right child and its children. This construction is similar to the one outlined in [311]. Also,  $\mathcal{T}_0$  corresponds to the defective node. Since the random walk is biased towards the minimizer, therefore given the construction of  $\mathcal{T}_i$ , the probability that random walk is still in one of such sub-trees would decrease with time. To formalize this idea, we consider the last passage times of any sub-tree  $\mathcal{T}_i$  for  $1 \leq i \leq \lceil \log_2 n \rceil$ . Let  $\tau_i$  denote the last passage time to  $\mathcal{T}_i$ .

We are interested in evaluating the last passage time to a sub-tree  $\tau_i$ . Note that all of them are distributed identically and WLOG we just consider  $\tau_{\lceil \log_2 n \rceil}$ . Now, this problem of random walk on  $\mathcal{T}_{\lceil \log_2 n \rceil}$  can be mapped to the problem of a random walk on the set  $S = \{-1, 0, 1, 2, \dots, \lceil \log_2 n \rceil\}$ . If each non-negative integer is mapped to the subset of nodes at the corresponding depth in the sub-tree,

then our random walk on  $\mathcal{T}_{\lceil \log_2 n \rceil}$  between different levels is equivalent to the random walk on these integers. Note that since the defective not contained in this sub-tree, all nodes at the same depth are identical in terms of distance to the defective, or more specifically, are equally far away from exiting the tree and therefore are can be abstracted into single node. This abstraction is precisely what leads to the equivalence between the two problems. Under this setup, the root node is mapped to 0 and the left sub-tree of the root node is mapped to  $-1$ , indicating an exit from the sub-tree  $\mathcal{T}_{\lceil \log_2 n \rceil}$ .

To analyze the random walk on the set of integers, we model it as a Markov Chain on the  $S$ , where  $\mathbb{P}(j \rightarrow j + 1) = p$  for all  $j \in S$ ,  $\mathbb{P}(j \rightarrow j - 1) = 1 - p$  for all  $j \geq 0$  and  $\mathbb{P}(-1 \rightarrow -1) = 1 - p$ . The probability  $1 - p$  is same as the bias of the random walk. The initial state of this Markov Chain is 0. Since under this model the arrival on the integer  $-1$  is equivalent to exiting the sub-tree  $\mathcal{T}_1$  in RWT, therefore, the event that the random walk in has not exited  $\mathcal{T}_1$  after  $n$  steps is the same as the Markov Chain being in a state  $j$  for  $j \geq 0$  after  $n$  steps. This event can be equally stated as the event that the number of steps taken by the random walk in the positive direction are at least as many as those taken in the negative direction since the random walk was initialized at 0. Combining all these ideas along with noting the specific structure of the transition matrix, we can conclude that

$$\mathbb{P}(\tau_{\lceil \log_2 n \rceil} > n) = \mathbb{P}(Z \leq n/2), \quad (\text{G.1})$$

where  $Z \sim \text{Bin}(n, 1 - p)$ . Writing expectation as the sum of tail probabilities,

$$\begin{aligned}
\mathbb{E}[\tau_{\lceil \log_2 n \rceil}] &= \sum_{n=0}^{\infty} \mathbb{P}(\tau_1 > n), \\
&= \sum_{n=0}^{\infty} \mathbb{P}(Z \leq n/2), \\
&= \sum_{n=0}^{\infty} \exp(-2(p - 1/2)^2 n), \\
&= \frac{1}{1 - \exp(-2(p - 1/2)^2)}. \tag{G.2}
\end{aligned}$$

The third step is obtained using Hoeffding's inequality. Using the definition of  $\tau_i$ , we can write that,

$$\begin{aligned}
\mathbb{E}[\tau_{RWT}(p, \epsilon)] &\leq \sum_{i=1}^{\lceil \log_2 n \rceil} \mathbb{E}[\tau_i](2N_0(p)) + \mathbb{E}[\tau_{\lceil \log_2 n \rceil + 1}] \\
&\leq \left( \frac{2N_0(p)}{1 - \exp(-2(p - 1/2)^2)} \right) (\log_2 n + 1) + \mathbb{E}[\tau_{\lceil \log_2 n \rceil + 1}] \tag{G.3}
\end{aligned}$$

Now, we know that the algorithm carries out the leaf level test at with error probability at most  $\delta$ . This implies that  $\mathbb{E}[\tau_{\lceil \log_2 n \rceil + 1}] = N_0(\delta)$ . We have to set  $\delta$  such that it ensures that the total probability of error is no more than  $\epsilon$ . Note that whenever the random walk arrives at a non-leaf target node, it will perform the leaf level test and the probability of it being accepted is  $\delta$ . Therefore, if  $\tau_{\text{leaf}}$  is the random number of times that a random walk arrives at a non-target leaf node, then the conditional probability of error is upper bounded by  $\delta\tau_{\text{leaf}}$ . From the definition of  $\tau_i$ 's,  $\tau_{\text{leaf}}$  is upper bounded by the  $\sum_{i=1}^{\lceil \log_2 n \rceil} \tau_i$ . Taking expectation, we get that the probability of error is upper bounded by

$$\Pr(\text{err}) \leq \delta \left( \frac{1}{1 - \exp(-2(p - 1/2)^2)} \right) (\log_2 n + 1)$$

Therefore, we set  $\delta = \frac{\epsilon}{\beta(\log_2 n + 1)}$  where  $\beta = \frac{1}{1 - \exp(-2(p - 1/2)^2)}$ . Combining this with Eqn. (G.3), we get the required bound.

Note that this analysis carries through almost as is even for the case when there are  $d$  defectives in the group. Since each of the  $d$  defectives are identified by a different run of the random walk, the expected number of samples required are simply  $d$  times the expected number of samples to identify a single defective. However, we also need to appropriately modify the error probability to ensure that the overall error for identifying all the  $d$  defectives is less than the prescribed threshold of  $\epsilon$ . Upon combining these ideas, we can conclude that the expected number of samples required to identify  $d$  defectives, denoted by  $\mathbb{E}[\tau_{RWT}^d(p, \epsilon)]$ , is upper bounded as

$$\mathbb{E}[\tau_{RWT}^d(p, \epsilon)] \leq 2d\beta(h + 1)N_0(p) + N_0(\epsilon'/d).$$

### G.3 Proof of Theorem 8.4.1

We begin with the proof for the expected sample complexity of the inner module. Note that the inner module simply involves carrying out the random walk with a bias of  $1 - p_{in}$  on a tree with  $n = d^2$  leaves, or equivalently, a depth of  $2 \log_2 n$  to identify at most  $d$  defectives with error probability of no greater than  $p_{out}$ . Moreover, this random walk is guided by the sequential test described in Section 8.3.3 whose sample complexity is characterized in Lemma 8.4.2. Thus, a direct application of Lemma 8.4.3 with the aforementioned parameters along with the result in Lemma 8.4.2 yields the following bound

$$\begin{aligned} \mathbb{E}[\tau_{in}(d, p_{in}, p_{out})] &\leq (2\beta d) \left( \frac{40}{\rho^2} \log \left( \frac{12}{\sqrt{p_{in}}} \log \left( \frac{240}{\sqrt{p_{in}\rho^2}} \right) \right) + 2 \right) (\log_2(d^2) + 1) \\ &\quad + d \left( \frac{40}{\rho^2} \log \left( 12 \sqrt{\frac{\beta d(\log_2(d^2) + 1)}{p_{out}}} \log \left( \frac{240}{\rho^2} \sqrt{\frac{\beta d(\log_2(d^2) + 1)}{p_{out}}} \right) \right) + 2 \right) \end{aligned}$$

where  $\beta = \frac{1}{1 - \exp(-2(p_{in} - 1/2)^2)}$ , as required.

The proof for the bound on  $\mathbb{E}[\tau_{HRW}(d, n, \epsilon)]$  is based on the the proof of Lemma 8.4.3. We draw parallels between the random walk carried out in the outer module and that in the classical setup of a binary tree which allows us to adopt the results from the latter, with minor modifications, for the analysis of the former.

In the outer module, since the random walk is induced on a general graph instead of a tree, the path joining the root node to the target node is not necessarily unique. Despite the non-uniqueness of the path, we can define a set of subgraphs corresponding to any node for the outer module that correspond to the subtrees  $\mathcal{T}_i$ 's defined in Appendix G.2. Let  $v$  be a node such that it is either a target node or the parent of a target node. Recall that the random walk in the outer module can zoom only into a proper subset of a given node. Consequently, the node  $v$  will lie on a path joining the root node to the target node. For any such node  $v$ , one can define a sub-graph  $\mathcal{G}_v$  that is the equivalent of sub-tree  $\mathcal{T}_i$ . Specifically, we define  $\mathcal{G}_v$  to be the sub-graph that contains the current node  $v$  along with all its children that are neither the parent of the target node nor the target node itself. Similar to  $\mathcal{T}_i$ 's, the subgraphs  $\mathcal{G}_v$ 's capture that part of the graph that corresponds to the incorrect direction of the random walk. Since the walk is biased, we can bound the time spent by the random walk in any such graph, using the same approach as used in the proof of Lemma 8.4.3. Therefore, for any such node  $v$ , we can bound the expected time spent in  $\mathcal{G}_v$  by a constant

$$\gamma = \frac{1}{1 - \exp(-2(p_{out} - 1/2)^2)}.$$

We can now carry out the same analysis as done in Appendix G.2 to bound the number of expected samples taken by the algorithm. Specifically, by substituting  $\mathcal{T}_i$ 's with  $\mathcal{G}_v$ 's,  $\beta$  with  $\gamma$  and using the  $\mathbb{E}[\tau_{in}(d, p_{in}, p_{out})]$  as the expected

sample complexity of the local test guiding the random walk, we can use the result in Lemma 8.4.3 to obtain a bound on  $\mathbb{E}[\tau_{QGT}(d, n, \epsilon)]$ . The only thing left to specify is equivalent of the “depth” of the tree, which in other words, corresponds to the length of the path between the root node and the target leaf node. Note that by the construction of the inner routine, if a current node represents a group of  $m$  items then it will zoom into a node with at most  $m/d + d$  items. Consequently, the corresponding “depth” for the random walk from the root to the target node is no more than  $\log_d n + 1$ . On combining this observation with the aforementioned analogy and the result of Lemma 8.4.3, we can conclude that

$$\mathbb{E}[\tau_{QGT}(d, n, \epsilon)] \leq \gamma(\log_d(n) + 1)\mathbb{E}[\tau_{in}(d, p_{in}, p_{out})] + \frac{40}{\rho^2} \log\left(\frac{12}{\sqrt{\epsilon_1}} \log\left(\frac{240}{\sqrt{\epsilon_1}\rho^2}\right)\right) + 2$$

where  $\gamma = \frac{1}{1 - \exp(-2(p_{out} - 1/2)^2)}$  and  $\epsilon_1 = \frac{\epsilon}{\gamma(\log_d n + 1)}$ . Note that the choice of  $\epsilon_1$  is made exactly in the same way as in Appendix G.2.

APPENDIX H  
CHAPTER 9

## H.1 Proof of Theorem 9.2.1

The central idea of this proof is to establish bounds on the moment generating function of the  $K_1$  statistic. Once we have obtained the bound on MGF, the final result follows directly by an application of Markov's inequality and union bound. Hence, we first focus on bounding the MGF of  $K_1$  and then would work on establishing bounds on probability of error and sample complexity.

Note that  $K_1$  is a sub-Gaussian random variable since it is bounded, so we can obtain an upper bound on its moment generating function using this property. However, the variance proxy obtained by just using the boundedness of  $K_1$  is not tight enough for our purposes. Thus, in order to obtain tight bounds on the variance proxy we employ an approach similar to the one used in Huang and Meyn [142]. Since their focus was mainly on asymptotic analysis of the coincidence test for obtained error rates, we need to appropriately modify the approach to obtain bounds for finite sample regime.

Before we begin the proof, we set up some additional notation that we will be using throughout the proof. The underlying discrete distribution is denoted by  $p$ . If  $\mathcal{S}$  denotes a set of i.i.d. samples from  $p$  such that  $|\mathcal{S}| = n$ , then with a slight abuse of notation, we denote  $K_1(\mathcal{S})$  as  $K_1(n)$  since the relevant quantity throughout the analysis will be the size of the sample. We derive the bounds for a general  $n \geq \sqrt{m}$  number of samples, followed by substituting the particular choices later. Lastly, we assume that  $n/m \leq \varepsilon^2/1536$ . While the sparse regime is

required to use the coincidence test, the additional dependence on  $\varepsilon$  is a result of the particular technique being employed to bound the MGF. We conjecture that requirement can be relaxed by using better analysis tools and techniques.

From Huang and Meyn [142, Eqn. 38], we know that the MGF of  $K_1$  satisfies the following equation:

$$\mathbb{E}_p \left[ \exp(\theta \tilde{K}_1(n)) \right] = e^{-\theta n} \frac{n!}{2\pi i} \oint g(\lambda) d\lambda, \quad (\text{H.1})$$

where  $\tilde{K}_1(n) = K_1(n) - n$  and

$$g(\lambda) = e^\lambda \prod_{j=1}^m \left( 1 - \lambda p_j e^{-\lambda p_j} + \lambda p_j e^{-\lambda p_j} e^\theta \right) \frac{1}{\lambda^{n+1}}.$$

Throughout the analysis, we assume that  $\theta \in [-0.4, 0.4]$ . The integral in Eqn. (H.1) is estimated using the saddle point method [87]. This approach consists of two steps. In the first step, a particular contour around  $\lambda = 0$  is chosen to carry out the integration. The choice of this contour is such that  $g(\lambda)$  behaves violently along it, i.e.,  $g(\lambda)$  is very large for a very small interval and significantly smaller on the rest of the contour, similar to a dirac delta function. This violent behaviour allows one to approximate the integral by only evaluating it on the small interval where the function value is very large. Such a contour is usually obtained by identifying a saddle point of  $g(\lambda)$ , a point where the derivative of  $g$  goes to zero and choosing to contour to pass through this point. The second step involves estimating the integral along the contour using the Laplace method.

To choose the contour, we first differentiate  $g$  to find the saddle point. On differentiating, we obtain,

$$g'(\lambda) = g(\lambda) \left( \sum_{j=1}^m \left( \frac{p_j(e^\theta - 1 + e^{\lambda p_j})}{\lambda p_j(e^\theta - 1) + e^{\lambda p_j}} \right) - \frac{n+1}{\lambda} \right).$$

Instead of exactly choosing the minimizer, we choose a point very close to it, defined as the solution to the following equation:

$$\frac{\lambda p_j(e^\theta - 1 + e^{\lambda p_j})}{\lambda p_j(e^\theta - 1) + e^{\lambda p_j}} = n. \quad (\text{H.2})$$

It is established in [142] that the above equation has a unique non-negative solution, which we denote using  $\lambda_0$ . Furthermore, it satisfies the following inequalities:

$$\begin{aligned} \text{For } \theta \geq 0 : \quad ne^{-\theta} &\leq \lambda_0 \leq n(1 + e^{-1}(e^\theta - 1)) \\ \text{For } \theta < 0 : \quad n(1 + e^{-1}(e^\theta - 1)) &\leq \lambda_0 \leq ne^{-\theta}. \end{aligned} \quad (\text{H.3})$$

From the above inequalities, it is clear that  $\lambda_0 = O(n)$ . We can also obtain a more refined approximation of the relation between  $\lambda_0$  and  $n$  as follows. We first split the set of symbols into two set based on the magnitude of their probabilities and obtain the approximation separately for these sets. In particular, we define  $\mathcal{W} = \{j : p_j \geq 8/m\}$ . Using this, we define  $\beta(p) = \sum_{j \in \mathcal{W}} p_j$ .

We first focus on the symbols that not do not belong in  $\mathcal{W}$ , i.e.,  $j \notin \mathcal{W}$ . For these symbols, the following set of inequalities hold.

$$\lambda_0 p_j e^\theta + \lambda_0^2 p_j^2 (1 - e^{2\theta}) + l(\theta) \lambda_0^3 p_j^3 \leq \frac{\lambda_0 p_j (e^\theta - 1 + e^{\lambda_0 p_j})}{\lambda_0 p_j (e^\theta - 1) + e^{\lambda_0 p_j}} \leq \lambda_0 p_j e^\theta + \lambda_0^2 p_j^2 (1 - e^{2\theta}) + u(\theta) \lambda_0^3 p_j^3, \quad (\text{H.4})$$

where  $l(\theta)$  and  $u(\theta)$  are the following functions of  $\theta$ :

$$l(\theta) = \begin{cases} \frac{1}{6} (6e^{3\theta} - 9e^\theta + 3) & \text{if } \theta < 0 \\ 0 & \text{if } \theta \geq 0 \end{cases}$$

$$u(\theta) = \begin{cases} 0 & \text{if } \theta < 0 \\ \frac{1}{6} (6e^{3\theta} - 9e^\theta + 3) & \text{if } \theta \geq 0 \end{cases}.$$

For the symbols in  $\mathcal{W}$ , we have,

$$\frac{\lambda_0 p_j(e^\theta - 1 + e^{\lambda_0 p_j})}{\lambda_0 p_j(e^\theta - 1) + e^{\lambda_0 p_j}} = D_j \lambda_0 p_j e^\theta,$$

$$\text{where } D_j = \frac{e^{-\theta} + e^{-\lambda_0 p_j}(1 - e^{-\theta})}{1 + \lambda_0 p_j e^{-\lambda_0 p_j}(e^\theta - 1)}.$$

On plugging these approximations for different regimes in Eqn. (H.2), we obtain that  $\lambda_0$  is of the form,  $\lambda_0 = \frac{ne^{-\theta}(1+w)}{1 + \sum_{j \in \mathcal{W}} p_j(D_j - 1)}$ , where  $w$  is the small approximating factor, on which we will obtain bounds. For brevity, we define  $D_* := 1 + \sum_{j \in \mathcal{W}} p_j(D_j - 1)$ . Lastly, we can obtain the range on the ratio  $(1+w)/D_*$  from the relations in Eqn. (H.3).

On summing the lower bound over  $j$  in Eqn. (H.4), we obtain,

$$\begin{aligned} n &\geq \sum_{j \in \mathcal{W}} D_j \lambda_0 p_j e^\theta + \sum_{j \notin \mathcal{W}} [\lambda_0 p_j e^\theta + \lambda_0^2 p_j^2 (1 - e^{2\theta}) + l(\theta) \lambda_0^3 p_j^3] \\ n &\geq \lambda_0 D_* e^\theta + \lambda_0^2 \sum_{j \notin \mathcal{W}} p_j^2 (1 - e^{2\theta}) + l(\theta) \lambda_0^3 \sum_{j \notin \mathcal{W}} p_j^3 \\ n &\geq n(1+w) + \frac{n^2}{D_*^2} (1+w)^2 \sum_{j \notin \mathcal{W}} p_j^2 (e^{-2\theta} - 1) + l(\theta) n^3 \left( \frac{1+w}{D_*} \right)^3 \sum_{j \notin \mathcal{W}} p_j^3 \end{aligned}$$

Consequently, we have,

$$w \leq \frac{n}{D_*^2} (1+w)^2 \sum_{j \notin \mathcal{W}} p_j^2 (1 - e^{-2\theta}) - l(\theta) n^2 \left( \frac{1+w}{D_*} \right)^3 \sum_{j=1}^m p_j^3. \quad (\text{H.5})$$

Similarly, summing over the upper bound yields us the following relation.

$$\begin{aligned} n &\leq \sum_{j \in \mathcal{W}} D_j \lambda_0 p_j e^\theta + \sum_{j \notin \mathcal{W}} [\lambda_0 p_j e^\theta + \lambda_0^2 p_j^2 (1 - e^{2\theta}) + u(\theta) \lambda_0^3 p_j^3] \\ n &\leq \lambda_0 D_* e^\theta + \lambda_0^2 \sum_{j \notin \mathcal{W}} p_j^2 (1 - e^{2\theta}) + u(\theta) \lambda_0^3 \sum_{j \notin \mathcal{W}} p_j^3 \\ n &\leq n(1+w) + \frac{n^2}{D_*^2} (1+w)^2 \sum_{j \notin \mathcal{W}} p_j^2 (e^{-2\theta} - 1) + u(\theta) n^3 \left( \frac{1+w}{D_*} \right)^3 \sum_{j \notin \mathcal{W}} p_j^3 \end{aligned}$$

Consequently, we have,

$$w \geq \frac{n(1+w)^2}{D_*^2} \left( \sum_{j \notin W} p_j^2 \right) (1 - e^{-2\theta}) - u(\theta) n^2 \left( \frac{1+w}{D_*} \right)^3 \sum_{j \notin W} p_j^3. \quad (\text{H.6})$$

With these relations at our disposal, we now move on to evaluating the integral in Eqn. (H.1) along the closed contour  $\lambda = \lambda_0 e^{i\psi}$  for  $\psi \in [-\pi, \pi]$ . The integral reads as

$$\begin{aligned} \mathbb{E}_p \left[ \exp \left( \theta \tilde{S}_n^* \right) \right] &= e^{-\theta n} \frac{n!}{2\pi} \int_{-\pi}^{\pi} g(\lambda_0 e^{i\psi}) \lambda_0 e^{i\psi} d\psi \\ &= \frac{n!}{2\pi} \lambda_0^{-n} e^{-\theta n} \operatorname{Re} \left[ \int_{-\pi}^{\pi} h(\psi) d\psi \right], \end{aligned}$$

where

$$h(\psi) = e^{-in\psi} \prod_{j=1}^m \left( \lambda_0 p_j (e^\theta - 1) e^{i\psi} + e^{\lambda_0 p_j e^{i\psi}} \right).$$

Instead of  $h(\psi)$ , we focus on bounding  $H(\psi) = \log(h(\psi))$  since it is easier to deal with sums than products. We have,

$$H(\psi) = -in\psi + \sum_{j=1}^m \log \left( \lambda_0 p_j (e^\theta - 1) e^{i\psi} + e^{\lambda_0 p_j e^{i\psi}} \right).$$

We split the integral into three parts by evaluating it over three different ranges, i.e.,  $[-\pi, -\pi/2]$ ,  $[-\pi/2, \pi/2]$  and  $(\pi/2, \pi]$ . We first consider the integral over  $[-\pi/2, \pi/2]$ . This is the region where the integrand behaviour behaves violently and thus will correspond to the dominant term. We have,

$$\begin{aligned} \operatorname{Re}(H(\psi)) &= \sum_{j=1}^m \operatorname{Re} \left[ \log \left( \lambda_0 p_j (e^\theta - 1) e^{i\psi} + e^{\lambda_0 p_j e^{i\psi}} \right) \right] \\ &= \sum_{j=1}^m \operatorname{Re} \left[ \log(e^{\lambda_0 p_j e^{i\psi}}) + \log \left( 1 + \lambda_0 p_j (e^\theta - 1) e^{i\psi} e^{-\lambda_0 p_j e^{i\psi}} \right) \right] \\ &\leq \sum_{j=1}^m \left[ \lambda_0 p_j \cos(\psi) + \log \left( 1 + \lambda_0 p_j e^{-\lambda_0 p_j \cos(\psi)} (e^\theta - 1) \right) \right]. \end{aligned}$$

We define a function  $G(\psi; u)$  that corresponds to the general form of the RHS in the above expression. That is for  $u \geq 0$ ,

$$G(\psi; u) = u \cos(\psi) + \log\left(1 + ue^{-u \cos(\psi)}(e^\theta - 1)\right).$$

Note that  $G'(0; u) = 0$  and  $G''(0; u) \leq -0.4u$  for  $\psi \in [-\pi/2, \pi/2]$ . Thus, by using mean value theorem it can be shown that  $G(\psi; u)$  satisfies  $G(\psi; u) \leq G(0; u) - 0.2u\psi^2$  for  $\psi \in [-\pi/2, \pi/2]$  for all  $u \geq 0$ . On plugging this inequality in previous one, we have,

$$\begin{aligned} \operatorname{Re}(H(\psi)) &\leq \sum_{j=1}^m G(\psi; \lambda_0 p_j) \\ &\leq \sum_{j=1}^m \left[ G(0; \lambda_0 p_j) - 0.2\lambda_0 p_j \psi^2 \right] \\ &\leq \sum_{j=1}^m \left[ \lambda_0 p_j + \log\left(1 + \lambda_0 p_j e^{-\lambda_0 p_j}(e^\theta - 1)\right) \right] - 0.2\lambda_0 \psi^2 \\ &\leq H(0) - 0.2\lambda_0 \psi^2. \end{aligned}$$

Using this relation, we can establish the following bound.

$$\begin{aligned} \operatorname{Re}\left[\int_{-\pi/2}^{\pi/2} h(\psi) d\psi\right] &= \operatorname{Re}\left[\int_{-\pi/2}^{\pi/2} \exp(H(\psi)) d\psi\right] \\ &\leq \int_{-\pi/2}^{\pi/2} |\exp(H(\psi))| d\psi \\ &\leq \int_{-\pi/2}^{\pi/2} \exp(\operatorname{Re}(H(\psi))) d\psi \\ &\leq e^{H(0)} \int_{-\pi/2}^{\pi/2} e^{-0.2\lambda_0 \psi^2} d\psi \\ &\leq e^{H(0)} \int_{-\infty}^{\infty} e^{-0.2\lambda_0 \psi^2} d\psi \\ &\leq e^{H(0)} \sqrt{\frac{\pi}{0.1\lambda_0}}. \end{aligned} \tag{H.7}$$

For  $\psi \in [-\pi, -\pi/2] \cup [\pi/2, \pi]$ , we have  $|e^{\lambda_0 p_j e^{i\psi}}| \leq 1$ . Consequently,  $|\lambda_0 p_j(e^\theta -$

$|1)e^{i\psi} + e^{\lambda_0 p_j e^{i\psi}}| \leq 1 + \lambda_0 p_j (e^\theta - 1)$ . Thus, we have,

$$\begin{aligned}\operatorname{Re}(H(\psi)) &= \sum_{j=1}^m \operatorname{Re} [\log (\lambda_0 p_j (e^\theta - 1) e^{i\psi} + e^{\lambda_0 p_j e^{i\psi}})] \\ &\leq \sum_{j=1}^m \log (1 + \lambda_0 p_j |(e^\theta - 1)|) \\ &\leq \sum_{j=1}^m \lambda_0 p_j |(e^\theta - 1)| \\ &\leq \lambda_0 |(e^\theta - 1)|.\end{aligned}$$

Using this bound, we can bound the following integral.

$$\begin{aligned}\operatorname{Re} \left[ \int_{-\pi}^{-\pi/2} h(\psi) d\psi \right] &= \operatorname{Re} \left[ \int_{-\pi}^{-\pi/2} \exp(H(\psi)) d\psi \right] \\ &\leq \int_{-\pi}^{-\pi/2} |\exp(H(\psi))| d\psi \\ &\leq \int_{-\pi}^{-\pi/2} \exp(\operatorname{Re}(H(\psi))) d\psi \\ &\leq \int_{-\pi}^{-\pi/2} e^{\lambda_0 |e^\theta - 1|} d\psi \\ &\leq \frac{\pi}{2} e^{\lambda_0 |e^\theta - 1|}.\end{aligned}$$

We can similarly bound the integral for  $[\pi/2, \pi]$ . On combining all of them, we have,

$$\begin{aligned}\mathbb{E}_p [\exp(\theta \tilde{K}_1(n))] &= \frac{n!}{2\pi} \lambda_0^{-n} e^{-\theta n} \operatorname{Re} \left[ \int_{-\pi}^{\pi} h(\psi) d\psi \right] \\ &\leq \frac{n!}{2\pi} \lambda_0^{-n} e^{-\theta n} \left( e^{H(0)} \sqrt{\frac{\pi}{0.1 \lambda_0}} + \pi e^{\lambda_0 |(e^\theta - 1)|} \right) \\ &\leq \frac{n!}{2\pi n^n} \left( \frac{D_*}{1+w} \right)^n \left( e^{H(0)} \sqrt{\frac{\pi}{0.1 \lambda_0}} + \pi e^{\lambda_0 |(e^\theta - 1)|} \right).\end{aligned}\tag{H.8}$$

Since  $\theta \in [-0.4, 0.4]$ , the second term can be upper bounded as  $e^{\lambda_0 |(e^\theta - 1)|} \leq e^{0.7n}$ .

Using the above bound on the moment generating function, we can evaluate bounds on the probability of error. We begin with bounding the probability of false alarm, that is, the underlying distribution is uniform and we fail to detect

it. For this scenario, the following relation holds for  $\theta < 0$ , where  $\tau_n$  denotes the threshold when  $n$  samples have been taken and  $P_e(n)$  denotes the error probability at that time instant.

$$\begin{aligned}
P_e(n) &= \Pr(K_1(n) - \mathbb{E}_u[K_1(n)] < -\tau_n) \\
&= \Pr(\theta(K_1(n) - \mathbb{E}_u[K_1(n)]) > -\theta\tau_n) \\
&= \Pr(\exp(\theta(K_1(n) - \mathbb{E}_u[K_1(n)])) > \exp(-\theta\tau_n)) \\
&\leq \mathbb{E} [\exp(\theta(K_1(n) - \mathbb{E}_u[K_1(n)]))] e^{\theta\tau_n} \\
&\leq \mathbb{E} [\exp(\theta(K_1(n) - n))] \exp(\theta(n - \mathbb{E}_u[K_1(n)] + \tau_n)) \\
&\leq \mathbb{E} [\exp(\theta\tilde{K}_1(n))] \exp\left(\theta\tau_n + \theta\left(\frac{n(n-1)}{m} - \frac{n}{2}\left(\frac{n-1}{m}\right)^2\right)\right) \\
&\leq \frac{n!}{2\pi n^n} \left(\frac{D_*}{1+w}\right)^n \left(e^{H(0)} \sqrt{\frac{\pi}{0.1\lambda_0}} + \pi e^{0.7n}\right) \exp\left(\theta\tau_n + \theta\left(\frac{n(n-1)}{m} - \frac{n}{2}\left(\frac{n-1}{m}\right)^2\right)\right) \\
&\leq \frac{n!e^n}{2\pi n^n} \left(\frac{D_*}{1+w}\right)^n \left(e^{H(0)-n} \sqrt{\frac{\pi}{0.1\lambda_0}} + \pi e^{-0.3n}\right) \exp\left(\theta\tau_n + \theta\left(\frac{n(n-1)}{m} - \frac{n}{2}\left(\frac{n-1}{m}\right)^2\right)\right)
\end{aligned}$$

Since the underlying distribution is uniform,  $\mathcal{W} = \emptyset$  and consequently  $D_* =$

1. The expression  $H(0)$  can be bounded as follows for  $\theta < 0$ .

$$\begin{aligned}
H(0) &= \sum_{j=1}^m \log(\lambda_0 p_j(e^\theta - 1) + e^{\lambda_0 p_j}) \\
&\leq \sum_{j=1}^m \left[ \lambda_0 p_j e^\theta + \frac{1}{2} \lambda_0^2 p_j^2 (1 - e^{2\theta}) \right].
\end{aligned}$$

Using the above relation, we first bound the first term, which we denote by

$T_1$ , as that is the dominant term in the expression. We have,

$$\begin{aligned}
T_1 &= \left( \frac{D_*}{1+w} \right)^n \exp \left( H(0) - n + \theta \tau_n + \theta \left( \frac{n(n-1)}{m} - \frac{n}{2} \left( \frac{n-1}{m} \right)^2 \right) \right) \\
&\leq (1+w)^{-n} \exp \left( \sum_{j=1}^m \left[ \lambda_0 p_j e^\theta + \frac{1}{2} \lambda_0^2 p_j^2 (1 - e^{2\theta}) \right] - n + \theta \tau_n + \theta \left( \frac{n(n-1)}{m} - \frac{n}{2} \left( \frac{n-1}{m} \right)^2 \right) \right) \\
&\leq (1+w)^{-n} \exp \left( \lambda_0 e^\theta + \frac{n^2}{2m} (1+w)^2 (e^{-2\theta} - 1) - n + \theta \tau_n + \theta \left( \frac{n(n-1)}{m} - \frac{n}{2} \left( \frac{n-1}{m} \right)^2 \right) \right) \\
&\leq \exp \left( \frac{n^2}{2m} (-2\theta + 4\theta^2) + \theta \tau_n + \theta \left( \frac{n(n-1)}{m} - \frac{n}{2} \left( \frac{n-1}{m} \right)^2 \right) + n(w - \log(1+w)) \right) \\
&\leq \exp \left( \frac{2n^2\theta^2}{m} + \theta \left( \tau_n - \frac{n}{m} - \frac{n}{2} \left( \frac{n}{m} \right)^2 \right) + 2nw^2 \right).
\end{aligned}$$

Using the condition on the ratio of  $n$  and  $m$ , we have

$$\begin{aligned}
\frac{n}{m} + \frac{n^3}{2m^2} &\leq \frac{n^3}{m^2} \\
&\leq \frac{n^2\epsilon^2}{1536m}.
\end{aligned}$$

Note that for the particular choice of  $\tau$  and corresponding decision instant in the Sequential Coincidence Test satisfies  $n^2\epsilon^2/1536m \leq \tau_n/75$  for all epochs  $k$ . Lastly, using Eqn. (H.5), we have,  $2nw^2 \leq 2n \left( 3.76 \frac{n}{m} (1 - e^{-2\theta}) \right)^2 \leq \frac{114n^2\epsilon^2\theta^2}{1536m}$ . On combining these two bounds with the bound on  $T_1$ , we obtain,

$$T_1 \leq \exp \left( \frac{n^2\theta^2}{m} \left( 2 + \frac{114\epsilon^2}{1536} \right) + \frac{74\theta\tau_n}{75} \right).$$

On plugging  $\theta = -\frac{74m\tau_n}{375n^2}$ , we obtain

$$T_1 \leq \exp \left( -\frac{5476}{56250} \cdot \frac{m\tau_n^2}{n^2} \right).$$

On evaluating the above expression at decision instant for epoch  $k$ , that is, at  $n = n_k$  and  $\tau = \tau_k$ , we obtain

$$\begin{aligned}
T_1 &\leq \exp(-3 \log(k+2/\delta)) \\
&\leq \frac{1}{(k+2/\delta)^3}.
\end{aligned}$$

Let us consider the second term of the expression with the above value of  $\theta$ , denoted by  $T_2$ . We have,

$$\begin{aligned} T_2 &= \left( \frac{D_*}{1+w} \right)^n \exp \left( -0.3n + \theta\tau_n + \theta \left( \frac{n(n-1)}{m} - \frac{n}{2} \left( \frac{n-1}{m} \right)^2 \right) \right) \\ &\leq \exp \left( -0.3n - \frac{n\theta}{2} \left( \frac{n-1}{m} \right)^2 - n \log(1+w) \right) \\ &\leq \exp(-0.25n), \end{aligned}$$

where the last expression follows using the value of  $\theta$  and bound on  $w$  obtained from Eqn. (H.6).

On combining both the expressions, we obtain the following result for the probability of error at the decision instant corresponding to epoch  $k$ .

$$\begin{aligned} P_e(n_k) &\leq \frac{n_k! e^{n_k}}{2\pi n_k^{n_k}} \left( \frac{1}{(k+2/\delta)^3} \cdot \sqrt{\frac{18\pi}{n_k}} + \pi e^{-0.25n_k} \right) \\ &\leq e^{1/12n_k} \left( \frac{3}{(k+2/\delta)^3} + \sqrt{\frac{n_k\pi}{2}} e^{-0.25n_k} \right) \end{aligned}$$

In the last step above we have used Stirling's approximation. Using the above expression for  $P_e(n_k)$ , we can obtain the probability for false alarm, which we

denote by  $\Pr(\text{err}|H_0)$ .

$$\begin{aligned}
\Pr(\text{err}|H_0) &= \Pr\left(\bigcup_{k=1}^{\kappa} \{\mathbb{E}_u[K_1(n_k)] - K_1(n_k) > \tau_k\}\right) \\
&\leq \sum_{k=1}^{\kappa} \Pr(K_1^{(n_k)} - \mathbb{E}_u[K_1^{(n_k)}] < -\tau_k) \\
&\leq \sum_{k=1}^{\kappa} P_e(n_k) \\
&\leq \sum_{k=1}^{\kappa} e^{1/12n_k} \left( \frac{3}{(k+2/\delta)^3} + \sqrt{\frac{n_k \pi}{2}} e^{-0.25n_k} \right) \\
&\leq \left(1 + \frac{1}{6\sqrt{m}}\right) \sum_{k=1}^{\kappa} \left( \frac{3}{(k+2/\delta)^3} + \sqrt{\frac{\sqrt{m}\pi}{2}} e^{-0.25\sqrt{m}} \right) \\
&\leq \left(1 + \frac{1}{6\sqrt{m}}\right) \left( 3 \int_0^\infty \frac{1}{(x+2/\delta)^3} dx + \sqrt{\frac{\sqrt{m}\pi}{2}} \frac{96e^{-0.25\sqrt{m}}}{\varepsilon^2} \right) \\
&\leq \left(1 + \frac{1}{6\sqrt{m}}\right) \left( 3 \left(\frac{\delta}{2}\right)^2 + \sqrt{\frac{\sqrt{m}\pi}{2}} \frac{112e^{-0.25\sqrt{m}}}{\varepsilon^2} \right),
\end{aligned}$$

where the fifth line follows by noting that  $e^x \leq 1 + 2x$  for  $x \leq 1$ ,  $\sqrt{x}e^{-x/4}$  is decreasing for  $x > 2$  and  $n_k \geq \sqrt{m}$ . On plugging in the lower bound for  $m$ , we obtain that the above expression is less than  $\delta$ .

Having obtained the bounds for false alarm, we now apply a similar process for bounding the probability of miss detection. In this case, the underlying distribution  $p$  belongs to  $C(\varepsilon)$  such that  $\|p - u\|_1 = \gamma > \varepsilon$ . For this scenario, the following relation holds for  $\theta \geq 0$ , with  $\tau_n$  and  $P_e(n)$  defined in a similar way as

the previous case.

$$\begin{aligned}
\Pr(\text{err}) &= \Pr(\mathbb{E}_u[K_1(n)] - K_1(n) < \tau_n) \\
&= \Pr(K_1(n) - \mathbb{E}_u[K_1(n)] > -\tau_n) \\
&= \Pr(\theta(K_1(n) - \mathbb{E}_u[K_1(n)]) > -\theta\tau_n) \\
&= \Pr(\exp(\theta(K_1(n) - \mathbb{E}_u[K_1(n)])) > \exp(-\theta\tau_n)) \\
&\leq \mathbb{E} [\exp(\theta(K_1(n) - \mathbb{E}_u[K_1(n)]))] e^{\theta\tau_n} \\
&\leq \mathbb{E} [\exp(\theta(K_1(n) - n))] \exp(\theta(n - \mathbb{E}_u[K_1(n)] + \tau_n)) \\
&\leq \mathbb{E} [\exp(\theta(K_1(n) - n))] \exp\left(\theta\left(\tau_n + \frac{n^2}{m}\right)\right) \\
&\leq \frac{n!e^n}{2\pi n^n} \left(\frac{D_*}{1+w}\right)^n \left(e^{H(0)-n} \sqrt{\frac{\pi}{0.1\lambda_0}} + \pi e^{-0.3n}\right) \exp\left(\theta\left(\tau_n + \frac{n^2}{m}\right)\right).
\end{aligned}$$

For this scenario,  $H(0)$  can be bounded as follows for  $\theta \geq 0$ .

$$\begin{aligned}
H(0) &= \sum_{j=1}^m \log(\lambda_0 p_j (e^\theta - 1) + e^{\lambda_0 p_j}) \\
&\leq \sum_{j \notin \mathcal{W}} \left[ \lambda_0 p_j e^\theta + \frac{1}{2} \lambda_0^2 p_j^2 (1 - e^{2\theta}) + \frac{\theta e^{3\theta}}{2} \lambda_0^3 p_j^3 \right] + \sum_{j \in \mathcal{W}} (\lambda_0 p_j e^\theta + \lambda_0 p_j (1 - e^{\lambda_0 p_j})(1 - e^\theta))
\end{aligned}$$

Once again, we focus on the first term in the bound of  $P_e(n)$ , which we denote by  $T'_1$ , as that is the dominant term. Using the relation on  $H(0)$  obtained from

above, we have,

$$\begin{aligned}
T'_1 &= \left( \frac{D_*}{1+w} \right)^n \exp \left( H(0) - n + \theta \left( \tau_n + \frac{n^2}{m} \right) \right) \\
&\leq \left( \frac{D_*}{1+w} \right)^n \exp \left( -n + \theta \left( \tau_n + \frac{n^2}{m} \right) \right) \times \\
&\quad \exp \left( \sum_{j \notin W} \lambda_0 p_j e^\theta + \frac{1}{2} \sum_{j \notin W} \lambda_0^2 p_j^2 (1 - e^{2\theta}) + \frac{\theta e^{3\theta}}{2} \sum_{j \notin W} \lambda_0^3 p_j^3 + \sum_{j \in W} (\lambda_0 p_j e^\theta + \lambda_0 p_j (1 - e^{\lambda_0 p_j}) (1 - e^\theta)) \right) \\
&\leq \left( \frac{D_*}{1+w} \right)^n \exp \left( -n + \theta \left( \tau_n + \frac{n^2}{m} \right) \right) \times \\
&\quad \exp \left( \lambda_0 e^\theta + \frac{1}{2} \sum_{j \notin W} \lambda_0^2 p_j^2 (1 - e^{2\theta}) + \frac{\theta e^{3\theta}}{2} \sum_{j \notin W} \lambda_0^3 p_j^3 + \sum_{j \in W} (\lambda_0 p_j (1 - e^{\lambda_0 p_j}) (1 - e^\theta)) \right) \\
&\leq \left( \frac{D_*}{1+w} \right)^n \exp \left( -n + \theta \left( \tau_n + \frac{n^2}{m} \right) \right) \times \\
&\quad \exp \left( \frac{n(1+w)}{D_*} + \frac{n^2}{2} \left( \frac{1+w}{D_*} \right)^2 \sum_{j \notin W} p_j^2 (e^{-2\theta} - 1) + \frac{32\theta n^3}{m^2} \left( \frac{1+w}{D_*} \right)^3 + \sum_{j \in W} (\lambda_0 p_j (1 - e^{\lambda_0 p_j}) (1 - e^\theta)) \right) \\
&\leq \exp \left( \frac{n^2}{2} \sum_{j \notin W} p_j^2 (e^{-2\theta} - 1) + \theta \left( \tau_n + \frac{n^2}{m} \right) + \frac{32\theta n^3}{m^2} \left( \frac{1+w}{D_*} \right)^3 \right) \times \\
&\quad \exp \left( n \left\{ \frac{(1+w)}{D_*} \left( 1 + \sum_{j \in W} p_j (1 - e^{\lambda_0 p_j}) (e^{-\theta} - 1) \right) - 1 - \log \left( \frac{1+w}{D_*} \right) \right\} \right).
\end{aligned}$$

We consider the second term on RHS separately, denoting it by  $J$ .

$$J = \frac{(1+w)}{1 + \sum_{j \in W} p_j (D_j - 1)} \left( 1 + \sum_{j \in W} p_j (1 - e^{\lambda_0 p_j}) (e^{-\theta} - 1) \right) - 1 - \log \left( \frac{1+w}{1 + \sum_{j \in W} p_j (D_j - 1)} \right).$$

To analyse the relation between the terms  $\sum_{j \in W} p_j (D_j - 1)$  and  $\sum_{j \in W} p_j (1 - e^{\lambda_0 p_j}) (e^{-\theta} - 1)$ , we define two functions  $D(x)$  and  $E(x)$  as follows:

$$\begin{aligned}
D(x) &:= \frac{e^{-\theta} + e^{-x}(1 - e^{-\theta})}{1 + xe^{-x}(e^\theta - 1)} \\
E(x) &:= (e^{-\theta} - 1)(1 - e^{-x}).
\end{aligned}$$

Then we have,  $D_j = D(\lambda_0 p_j)$  and  $E(\lambda_0 p_j) = (1 - e^{\lambda_0 p_j})(e^{-\theta} - 1)$ . It is not difficult to note that the function  $F(x) = \frac{D(x)-1}{E(x)}$  is decreasing for all  $x \geq 0$  and it satisfies the

relation  $1 \leq F(x) \leq 1 + e^\theta$ . Consequently, we have, for all  $x \geq 0$ ,  $(1 + e^\theta)E(x) \leq D(x) - 1 \leq E(x)$  since  $E(x) \leq 0$ . Thus, we have,

$$(1 + e^\theta) \sum_{j \in \mathcal{W}} p_j(1 - e^{\lambda_0 p_j})(e^{-\theta} - 1) \leq \sum_{j \in \mathcal{W}} p_j(D_j - 1) \leq \sum_{j \in \mathcal{W}} p_j(1 - e^{\lambda_0 p_j})(e^{-\theta} - 1).$$

If we let  $x = \sum_{j \in \mathcal{W}} p_j(D_j - 1)$ , then we can write  $J$  as

$$J(x) = \frac{(1 + w)(1 + \rho x)}{1 + x} - 1 - \log\left(\frac{(1 + w)}{1 + x}\right),$$

where  $\rho \in [(1 + e^\theta)^{-1}, 1]$ .

Since  $D(y) \geq e^{-2\theta}$  for all  $y \geq 0$ , the domain of  $x$  is given as  $x \in [-\beta(p)(1 - e^{-2\theta}), 0]$ .

As  $\beta(p) < 1$  and  $\theta < 0.4$ , the function  $J(x)$  is increasing throughout the domain of  $x$ . Consequently,  $J(x) \leq J(0) \leq w^2$ . Furthermore, over this domain, we can upper bound  $J(x)$  as  $J(x) = w^2 + 0.2x$ .

Lastly, since  $D(x)$  is a decreasing function  $D_j \leq D\left(\frac{8\lambda_0}{m}\right)$ . Once again, using a local linear approximation of  $D(x)$ , we have the upper bound  $D(x) \leq 1 + 0.7(e^{-\theta} - e^\theta)x$ . On combining everything, we obtain the following relation.

$$\begin{aligned} J &= \frac{(1 + w)}{1 + \sum_{j \in \mathcal{W}} p_j(D_j - 1)} \left( 1 + \sum_{j \in \mathcal{W}} p_j(1 - e^{\lambda_0 p_j})(e^{-\theta} - 1) \right) - 1 - \log\left(\frac{1 + w}{1 + \sum_{j \in \mathcal{W}} p_j(D_j - 1)}\right) \\ &\leq w^2 + 0.2 \sum_{j \in \mathcal{W}} p_j(D_j - 1) \\ &\leq w^2 + 0.2 \sum_{j \in \mathcal{W}} p_j \left( D\left(\frac{8\lambda_0}{m}\right) - 1 \right) \\ &\leq w^2 + 0.14 \sum_{j \in \mathcal{W}} p_j \frac{8\lambda_0}{m} (e^{-\theta} - e^\theta) \\ &\leq w^2 + 1.12\beta(p) \frac{n(1 + w)}{mD_*} (e^{-2\theta} - 1). \end{aligned}$$

On plugging this back into the bound for  $T'_1$ , we obtain,

$$\begin{aligned} T'_1 &\leq \exp\left(\frac{n^2}{2} \sum_{j \notin \mathcal{W}} p_j^2 (e^{-2\theta} - 1) + \theta\left(\tau_n + \frac{n^2}{m}\right) + \frac{32\theta n^3}{m^2} \left(\frac{1+w}{D_*}\right)^3\right) \times \\ &\quad \exp\left(n\left(w^2 + 1.12\beta(p)\frac{n(1+w)}{mD_*}(e^{-2\theta} - 1)\right) + nw^2\right) \\ &\leq \exp\left(\frac{n^2}{2m} \left(\sum_{j \notin \mathcal{W}} p_j^2 + 2\beta(p)\right)(e^{-2\theta} - 1) + \theta\left(\tau_n + \frac{n^2}{m}\right) + \frac{32\theta n^3}{m^2} \left(\frac{1+w}{D_*}\right)^3 + nw^2\right). \end{aligned}$$

Let  $p_j = \frac{1}{m} + \Delta_j$ . So  $\sum_{j=1}^m \Delta_j = 0$  and  $\sum_{j=1}^m |\Delta_j| = \gamma$ , the actual  $\ell_1$  distance to the uniform distribution. Using this, the first term can be written as,

$$\begin{aligned} \frac{n^2}{2m} \left(m \sum_{j \notin \mathcal{W}} p_j^2 + 2 \sum_{j \in \mathcal{W}} p_j\right) &\geq \frac{n^2}{2m} \left(m \sum_{j \notin \mathcal{W}} \left(\frac{1}{m} + \Delta_j\right)^2 + 2 \sum_{j \notin \mathcal{W}} \left(\frac{1}{m} + \Delta_j\right)\right) \\ &\geq \frac{n^2}{2m} \left(1 + m \sum_{j \notin \mathcal{W}} \Delta_j^2 + \frac{|\mathcal{W}|}{m}\right) \\ &\geq \frac{n^2}{2m} \left(1 + \frac{\gamma^2}{4}\right), \end{aligned}$$

where the last step follows from  $\sum_{j \notin \mathcal{W}} |\Delta_j| \geq \gamma/2$ . The bound on  $n/m$  and Eqn. (H.3) gives us the following relation

$$\frac{32\theta n^3}{m^2} \left(\frac{1+w}{D_*}\right)^3 \leq \frac{\theta n^2 \epsilon^2}{8m}.$$

Lastly using Eqn. (H.5) and Eqn. (H.3), we have,  $nw^2 \leq \frac{n^3}{D_*^4} (1+w)^4 \left(\sum_{j \notin \mathcal{W}} p_j^2\right)^2 (1 - e^{-2\theta})^2 \leq \theta^2 \cdot \frac{7n^2 \epsilon^2}{4m}$ . On plugging on these bounds in the bound for  $T'_1$ , we obtain,

$$\begin{aligned} T'_1 &\leq \exp\left(-\frac{n^2}{2m} \left(1 + \frac{\gamma^2}{4}\right) (2\theta - 4\theta^2) + \theta\left(\tau_n + \frac{n^2}{m}\right) + \frac{\theta n^2 \epsilon^2}{8m} + \frac{7\theta^2 n^2 \epsilon^2}{4m}\right) \\ &\leq \exp\left(\frac{2n^2 \theta^2}{m} \left(1 + \frac{\gamma^2}{4} + \frac{7\epsilon^2}{8}\right) - \theta\left(\frac{n^2}{m} \left(\frac{\gamma^2}{4} - \frac{\epsilon^2}{8}\right) - \tau_n\right)\right) \\ &\leq \exp\left(\frac{2n^2 \theta^2}{m} \left(1 + \frac{\gamma^2}{4} + \frac{7\epsilon^2}{8}\right) - \theta\left(\frac{n^2 \gamma^2}{8m} - \tau_n\right)\right) \end{aligned}$$

Note that the particular choice of  $\tau$  and corresponding decision instant as used in Alg. 20 satisfy  $\frac{n^2 \gamma^2}{8m} \geq 2\tau_n$  for all  $k \geq 112/\gamma^2$ . Thus, if we define  $k_0(\gamma) =$

$112/\gamma^2$ , then for all  $k \geq k_0(\gamma)$ , we have

$$T'_1 \leq \exp\left(\frac{4.25n^2\theta^2}{m} - \theta\tau_n\right).$$

On plugging  $\theta = \frac{m\tau_n}{9.5n^2}$ , we obtain,

$$T'_1 \leq \exp\left(-\frac{m\tau_n^2}{18n^2}\right).$$

On evaluating the above expression at decision instant for epoch  $k \geq k_0(\gamma)$ , that is, at  $n = n_k$  and  $\tau = \tau_k$ , we obtain

$$\begin{aligned} T'_1 &\leq \exp(-2.5 \log(k + 2/\delta)) \\ &\leq \frac{1}{(k + 2/\delta)^{2.5}}. \end{aligned}$$

Similar to the previous case, we us consider the second term of the expression, denoted by  $T'_2$  with the above value of  $\theta$ . Once again, we evaluate it at a decision instant for  $k \geq k(\gamma)$ . We have,

$$\begin{aligned} T_2 &= \left(\frac{D_*}{1+w}\right)^n \exp\left(-0.3n + \theta\left(\tau_n + \frac{n^2}{m}\right)\right) \\ &\leq \exp\left(-0.3n + \frac{17\theta n^2}{16m} - n \log((1+w)/D^*)\right) \\ &\leq \exp(-0.25n), \end{aligned}$$

where the last expression follows using the value of  $\theta$  and bound on  $w$  obtained from Eqn. (H.5).

On combining both the expressions, we obtain the following result for the probability of error at the decision instant corresponding to epoch  $k$ .

$$\begin{aligned} P_e(n_k) &\leq \frac{n_k! e^{n_k}}{2\pi n_k^{n_k}} \left( \frac{1}{(k+2/\delta)^{2.5}} \cdot \sqrt{\frac{18\pi}{n_k}} + \pi e^{-0.25n_k} \right) \\ &\leq e^{1/12n_k} \left( \frac{3}{(k+2/\delta)^{2.5}} + \sqrt{\frac{n_k\pi}{2}} e^{-0.25n_k} \right), \end{aligned}$$

where the last line again employs Stirling's Approximation.

Using the above expression for  $P_e(n_k)$ , we can obtain the probability for miss detection, which we denote by  $\Pr(\text{err}|H_1)$ .

$$\begin{aligned}
\Pr(\text{err}|H_1) &= \Pr\left(\bigcap_{k=1}^{\kappa} \{\mathbb{E}_u[K_1(n_k)] - K_1(n_k) < \tau_k\}\right) \\
&\leq \Pr\left(K_1^{(n_\kappa)} - \mathbb{E}_u[K_1^{(n_\kappa)}] > -\tau_\kappa\right) \\
&\leq P_e(n_\kappa) \\
&\leq e^{1/12n_\kappa} \left( \frac{3}{(\kappa + 2/\delta)^{2.5}} + \sqrt{\frac{n_\kappa \pi}{2}} e^{-0.25n_\kappa} \right) \\
&\leq \delta.
\end{aligned}$$

Thus, we have shown that probability of miss detection is also upper bounded by  $\delta$ . For the last part, we established the expected sample complexity of the routine. Let  $\Gamma$  denote the random number of sample taken by the procedure. For the scenario when the underlying distribution is uniform, we use a simple upper bound given as

$$\begin{aligned}
\mathbb{E}[\Gamma|H_0] &\leq n_\kappa \\
&\leq \frac{112\sqrt{m}}{\varepsilon^2} \sqrt{\log\left(\frac{112}{\varepsilon^2} + \frac{2}{\delta}\right)} + 1.
\end{aligned}$$

We now consider the case when the underlying distribution belongs to  $C(\varepsilon)$

such that  $\|p - u\|_1 = \gamma > \varepsilon$ . The expected sample complexity is given as

$$\begin{aligned}
\mathbb{E}[\Gamma|H_1] &= \sum_{k=1}^{\kappa} n_k \Pr(\Gamma = n_k) \\
&\leq n_{k_0(\gamma)} + \sum_{k=k_0(\gamma)+1}^{\kappa} n_k \Pr(\Gamma = n_k) \\
&\leq n_{k_0(\gamma)} + \sum_{k=k_0(\gamma)}^{\kappa} n_{k+1} \Pr(\Gamma > n_k) \\
&\leq n_{k_0(\gamma)} + \sum_{k=k_0(\gamma)}^{\kappa} n_{k+1} P_e(n_k) \\
&\leq n_{k_0(\gamma)} + \sum_{k=k_0(\gamma)}^{\kappa} n_{k+1} e^{1/12n_k} \left( \frac{3}{(k+2/\delta)^{2.5}} + \sqrt{\frac{n_k \pi}{2}} e^{-0.25n_k} \right) \\
&\leq n_{k_0(\gamma)} + e^{1/12} \sum_{k=k_0(\gamma)}^{\kappa} \left( \frac{4.5 \sqrt{m \log(k+2/\delta)}}{(k+2/\delta)^{1.5}} + \sqrt{\frac{9\pi}{8}} n_k^{1.5} e^{-0.25n_k} \right) \\
&\leq n_{k_0(\gamma)} + e^{1/12} \left( 4.5 \sqrt{m} \int_{k_0(\gamma)-1}^{\infty} \frac{\log(x+2/\delta)}{(x+2/\delta)^{1.5}} dx + \sqrt{\frac{9\pi}{8}} \sum_{k=k_0(\gamma)}^{\kappa} n_k^{1.5} e^{-0.25n_k} \right) \\
&\leq n_{k_0(\gamma)} + e^{1/12} \left( 27 \sqrt{m} \left( k_0(\gamma) - 1 + \frac{2}{\delta} \right)^{-1/2} \log \left( k_0(\gamma) - 1 + \frac{2}{\delta} \right) + \sqrt{\frac{9\pi}{8}} C_0 \delta \sqrt{m} \right),
\end{aligned}$$

for some universal constant  $C_0$ . The dominant term in the above expression is  $n_{k_0(\gamma)}$ , giving us,  $\mathbb{E}[\Gamma|H_1] = O\left(\gamma^{-2} \sqrt{m \log\left(\frac{1}{\gamma} + \frac{1}{\delta}\right)}\right)$ , as required.

## H.2 Proof of Theorem 9.3.1

The proof of this theorem largely builds on the proof of the previous theorem. The basic idea is to first show that for a fine enough discretization, the  $\ell_1$  distance of resulting discrete distribution is the same as that of the continuous distribution upto a constant factor. Once this relation is established, we can simply invoke the results obtained in the previous theorem to obtain the results. We begin with the following lemma that relates the  $\ell_1$  distances of the discrete and

continuous distributions.

**Lemma H.2.1.** *Let  $p$  be a distribution in  $C_L(\varepsilon)$  such that  $\|p - u\|_1 = \gamma$  and let  $p^\Delta$  be the discrete distribution obtained by a uniform discretization of the interval  $[0, 1]$  into  $m$  bins. The  $\ell_1$  distance of  $p^\Delta$  from the uniform distribution, denoted by  $[\gamma]_m$  satisfies the following relation*

$$[\gamma]_m = \sum_{i=1}^m |p_i^\Delta - 1/m| \geq \gamma - L/m,$$

where  $p_i^\Delta$  denotes the probability mass of  $p^\Delta$  in the  $i^{\text{th}}$  bin.

*Proof.* The proof of the result uses the Lipschitz continuity of the PDF of  $p$  to bound the error between the  $\ell_1$  distances of the continuous and the discrete distributions. We denote the continuous uniform distribution on  $[0, 1]$  using  $u(x)$ . From the definition of  $\ell_1$  distance, we have,

$$\begin{aligned} \gamma &= \int_0^1 |p(x) - u(x)| dx \\ &= \sum_{i=0}^{m-1} \int_{i/m}^{(i+1)/m} |p(x) - u(x)| dx \\ &= \sum_{i=0}^{m-1} \int_{i/m}^{(i+1)/m} |p(x) - mp_i^\Delta + mp_i^\Delta - u(x)| dx \\ &\leq \sum_{i=0}^{m-1} \int_{i/m}^{(i+1)/m} (|p(x) - mp_i^\Delta| + |mp_i^\Delta - 1|) dx \\ &\leq \sum_{i=0}^{m-1} \int_{i/m}^{(i+1)/m} \left( \frac{L}{m} + |mp_i^\Delta - 1| \right) dx \\ &\leq \sum_{i=0}^{m-1} \left[ \frac{L}{m^2} + \left| \int_{(i-1)/m}^{i/m} p(y) dy - \frac{1}{m} \right| \left( \int_{i/m}^{(i+1)/m} m dx \right) \right] \\ &\leq \sum_{i=0}^{m-1} \frac{L}{m^2} + \sum_{i=0}^{m-1} \left| p_i^\Delta - \frac{1}{m} \right| \\ &\leq \frac{L}{m} + [\gamma]_m. \end{aligned}$$

In the fifth step, we have used the mean value theorem along with the Lipschitz condition on the PDF. From the mean value theorem, we can conclude that there exists an  $x_i \in [i/m, (i+1)/m]$  such that  $p(x_i) = p_i^\Delta/(1/m) = mp_i$  for all  $i = 0, 1, 2, \dots, m-1$  and since PDF is  $L$ -Lipschitz, we have  $|p(x) - p(x_i)| \leq L/m$  for all  $x \in [i/m, (i+1)/m]$ . Consequently, we can relate the  $\ell_1$  distances between the continuous and discrete distributions. In particular, if  $m \geq L/2\gamma$ , then  $[\gamma]_m \geq \gamma/2$ .  $\square$

Once we have obtained a discretization, the proof for the probability of error and sample complexity is almost the same as that of previous case with very minor modifications. Firstly, we can simply modify the set  $\mathcal{W}$  to  $\mathcal{W}_k$  defined for each epoch as  $\mathcal{W}_k = \{j : p_j^\Delta \geq 8/m_k\}$ , where once again  $p_j^\Delta$  is the mass in the  $j^{\text{th}}$  bin in the discretization. Also for this case, instead of computing the error for any sample  $n$ , we just compute it for the decision instant  $n_k$  and the corresponding number of bins  $m_k$ .

Once again we begin with the probability of false alarm. It can be verified that for the choice of  $n_k$  and  $m_k$ , all the conditions in previous analysis are satisfied yielding us same bounds on  $T_1$  and  $T_2$  and consequently, the following result holds for the probability of error at the decision instant corresponding to epoch  $k$ .

$$P_e(n_k) \leq e^{1/12n_k} \left( \frac{3}{(k + 2/\delta)^3} + \sqrt{\frac{n_k \pi}{2}} e^{-0.25n_k} \right).$$

Using a similar process as in the previous proof, we can obtain the probability

for false alarm, which we denote by  $\Pr(\text{err}|H_0)$ .

$$\begin{aligned}
\Pr(\text{err}|H_0) &= \Pr\left(\bigcup_{k=1}^{\kappa} \{\mathbb{E}_u[K_1(n_k)] - K_1(n_k) > \tau_k\}\right) \\
&\leq \sum_{k=1}^{\kappa} \Pr(K_1^{(n_k)} - \mathbb{E}_u[K_1^{(n_k)}] < -\tau_k) \\
&\leq \sum_{k=1}^{\kappa} P_e(n_k) \\
&\leq \sum_{k=1}^{\kappa} e^{1/12n_k} \left( \frac{3}{(k+2/\delta)^3} + \sqrt{\frac{n_k \pi}{2}} e^{-0.25n_k} \right) \\
&\leq \delta.
\end{aligned}$$

The last line follows by the same reasoning as in the proof of the previous theorem.

We consider the probability of missed detection in a similar manner. In this scenario, the  $\ell_1$  distance to the uniform distribution of the underlying distribution  $p$  is  $\gamma$  and for simplicity, we denote the discretized  $\ell_1$  distance as  $[\gamma]_k$  instead of  $[\gamma]_{m_k}$ . If we define  $k_0(\gamma) = \min\{k : k \geq 144[\gamma]_k^{-2}\}$ , then using this definition of  $k_0(\gamma)$ , we can obtain all the results from the previous theorem. Specifically, for  $k \geq k_0(\gamma)$ , we have,

$$\begin{aligned}
\frac{32\theta n_k^3}{m_k^2} \left( \frac{1+w}{D_*} \right)^3 &\leq \frac{\theta n_k^2 [\gamma]_k^2}{8m_k}, \\
n_k w^2 &\leq \theta^2 \cdot \frac{7n_k^2 [\gamma]_k^2}{4m_k}, \\
2\tau_k &\leq \frac{n_k^2 [\gamma]_k^2}{8m_k}.
\end{aligned}$$

Consequently, for  $k \geq k_0(\gamma)$ , we have,

$$T'_1 \leq \exp\left(\frac{4.25n_k^2\theta^2}{m_k} - \theta\tau_k\right).$$

On plugging  $\theta = \frac{m_k \tau_k}{9.5 n_k^2}$ , we obtain,

$$\begin{aligned} T'_1 &\leq \exp\left(-\frac{m_k \tau_k^2}{18 n_k^2}\right) \\ &\leq \frac{1}{(k + 2/\delta)^{4.5}}. \end{aligned}$$

It is not difficult to see that  $T'_2 \leq \exp(-0.25n_k)$  for  $k \geq k_0(\gamma)$ , yielding a similar expression for  $P_e(n_k)$  and consequently we can conclude that  $\Pr(\text{err}|H_1) \leq \delta$ .

The expected sample complexity for the uniform case is simply bounded as  $n_\kappa$  implying that  $\mathbb{E}[\Gamma|H_0]$  is  $O(\varepsilon^{-6} \log(\varepsilon^{-1} + \delta^{-1}))$ . Lastly, the sample complexity for case when the underlying distribution belongs to  $C_L(\varepsilon)$  such that  $\|p - u\|_1 = \gamma > \varepsilon$  is given as follows.

$$\begin{aligned} \mathbb{E}[\Gamma|H_1] &= \sum_{k=1}^{\kappa} n_k \Pr(\Gamma = n_k) \\ &\leq n_{k_0(\gamma)} + \sum_{k=k_0(\gamma)+1}^{\kappa} n_k \Pr(\Gamma = n_k) \\ &\leq n_{k_0(\gamma)} + \sum_{k=k_0(\gamma)}^{\kappa} n_{k+1} \Pr(\Gamma > n_k) \\ &\leq n_{k_0(\gamma)} + \sum_{k=k_0(\gamma)}^{\kappa} n_{k+1} P_e(n_k) \\ &\leq n_{k_0(\gamma)} + \sum_{k=k_0(\gamma)}^{\kappa} n_{k+1} e^{1/12n_k} \left( \frac{3}{(k + 2/\delta)^{4.5}} + \sqrt{\frac{n_k \pi}{2}} e^{-0.25n_k} \right) \\ &\leq n_{k_0(\gamma)} + e^{1/12} \sum_{k=k_0(\gamma)}^{\kappa} \left( \frac{3 \sqrt{c_0} \log(k + 2/\delta)}{(k + 2/\delta)^{1.5}} + 7.5 \sqrt{\frac{\pi}{2}} n_k^{1.5} e^{-0.25n_k} \right) \\ &\leq n_{k_0(\gamma)} + e^{1/12} \left( 3 \sqrt{c_0} \int_{k_0(\gamma)-1}^{\infty} \frac{\log(x + 2/\delta)}{(x + 2/\delta)^{1.5}} dx + 7.5 \sqrt{\frac{\pi}{2}} \sum_{k=k_0(\gamma)}^{\kappa} n_k^{1.5} e^{-0.25n_k} \right) \\ &\leq n_{k_0(\gamma)} + e^{1/12} \left( 18 \sqrt{c_0} \left( k_0(\gamma) - 1 + \frac{2}{\delta} \right)^{-1/2} \log \left( k_0(\gamma) - 1 + \frac{2}{\delta} \right) + 7.5 \sqrt{\frac{\pi}{2}} C_1 \delta \sqrt{c_0} \right), \end{aligned}$$

for some universal constant  $C_1$ . The dominant term in the above expression is  $n_{k_0(\gamma)}$ . From the particular choice of  $m_k$  and Lemma H.2.1, we can conclude that

$k_0(\gamma) \leq 576\gamma^{-2}$ . This bound along with the expression for  $n_{k_0(\gamma)}$  yields  $\mathbb{E}[\Gamma|H_1] = O(\gamma^{-6} \log(\gamma^{-1} + \delta^{-1}))$ , as required.

ProQuest Number: 30528911

INFORMATION TO ALL USERS

The quality and completeness of this reproduction is dependent on the quality  
and completeness of the copy made available to ProQuest.



Distributed by ProQuest LLC (2023).

Copyright of the Dissertation is held by the Author unless otherwise noted.

This work may be used in accordance with the terms of the Creative Commons license  
or other rights statement, as indicated in the copyright statement or in the metadata  
associated with this work. Unless otherwise specified in the copyright statement  
or the metadata, all rights are reserved by the copyright holder.

This work is protected against unauthorized copying under Title 17,  
United States Code and other applicable copyright laws.

Microform Edition where available © ProQuest LLC. No reproduction or digitization  
of the Microform Edition is authorized without permission of ProQuest LLC.

ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 - 1346 USA