DEEP CODE: REPRESENTATION AND LEARNING ALGORITHMS FOR NEURAL NETWORKS & THEIR APPLICATIONS TO COMMUNICATION CODES

BY

ASHOK VARDHAN MAKKUVA

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2022

Urbana, Illinois

Abstract

Codes are the backbone of modern information age. Codes, composed of encoder and decoder pairs, are the basic mathematical objects that enable reliable communication. Landmark codes include convolutional, Reed-Muller, turbo, LDPC, and polar: each is linear and represents a mathematical breakthrough. Their impact on humanity is huge; each of these codes has been used in global communication standards over the past six decades. On the other hand, designing codes is a challenging task, mostly driven by human ingenuity. Befittingly, historically, the progress in discovery of codes has been sporadic.

In this thesis, we present a new paradigm to invent codes via harnessing tools from deep-learning. Our major result is the invention of *KO codes*, a computationally efficient family of deep-learning driven codes that outperform the state-of-the-art RM and polar codes, in the challenging short-to-medium block length regime. The key technical innovation behind KO codes is the design of a novel family of neural architectures inspired by the computation tree of the **K**ronecker **O**peration (KO) central to RM and polar codes. These architectures pave the way for discovery of a much richer class of hitherto unexplored non-linear codes. This design technique can be viewed as an instantiation of the classical neural augmentation principle. In the process, we also study a popular neural network model called Mixture-of-Experts (MoE) that realizes this principle. We provide the first set of consistent and efficient algorithms with global learning guarantees for learning the parameters in a MoE which has been an open problem for more than two decades.

To my family.

Acknowledgments

I would first like to thank my advisor Prof. Pramod Viswanath for his constant support and guidance throughout my PhD. Especially, I would like to thank him for helping me discover and nurture the creative spirit in me. As I progressed in grad school, thanks to him I began to view research more and more as a creative endeavour and less as a technical chore. This change in perspective has had such a huge impact on me and my research. I am greatly indebted to him for shining this light. His invaluable advice on various aspects of research – creating a problem statement and traversing the unexplored rugged research terrain with a torch called "your own co-ordinate system," identifying the right set of problems to work on, creating your vision, etc. – has greatly shaped my thought process and the perspective to view and understand things.

I have also had the pleasure to work with Prof. Sewoong Oh closely on a lot of research topics right from the beginning of my PhD. Working with Sewoong has helped me realize and inculcate many crucial aspects of high quality research: intuition, rigor, and clarity of thought. His ability to get to the core of the problem and digest complex ideas through intuition is one thing that I have always tried to imbibe into myself. Another close collaborator who has had tremendous influence on me is Prof. Sreeram Kannan. He taught me so many valuable things regarding becoming a world-class researcher. His energetic personality, endless curiosity, and being a generating source of numerous research problems has always been an inspiration to me. I am also deeply indebted to my amazing mentors from IITB, Prof. Vivek Borkar and Prof. Sibiraj Pillai, whose guidance and help carved the grad school path for me.

I would like to thank my doctoral committee members, together with Sewoong, Profs. Bruce Hajek, Rayadurgam Srikant, and Ruoyu Sun for their feedback and guidance. I would also like to thank Prof. Venu Veeravalli, Prof. Pierre Moulin, and Prof. Idoia Ochoa for serving as my qualifying exam committee. Special thanks to all the faculty whose courses have greatly shaped my thought process during the grad school: Prof. Richard Laugesen, Prof. Partha Dey, Prof. Maxim Raginsky, Prof. Zhongjin Ruan, Prof. Sewoong Oh, Prof. Matus Telgarsky, Prof. Ruoyu Sun, and Prof. Yihong Wu.

My life in Chambana would not have been as much fun and memorable if not for the amazing company of my friends across the years: Aniruddh, Anurendra, Anushree, Anirudh Chaudhary, Ishan, Kiran, Ashish, Arun, Tarek, Jason, Safa, Mohit, Manjunath, Pranjal, Sanket, Vipul, Suraj, Maghav, Moitreya, Tanmay, Deva, Srilakshmi, and Zeyu. As much as I deeply cherish the fun filled memorable times I spent with them, I would also like to thank them for being a source of constant source of inspiration and for providing myriad perspectives about life. Special thanks to Anand and Vedant, for being a living reminder that hard work and the right attitude can take you anywhere in life; Amir, for being a wonderful collaborator and a close friend; Anshika, for her passion for research and science; Anwesa and Maitreyee, for being an inspiration with their incredible multifaceted talents; Azin, for providing a lens to the fascinatingly beautiful Iranian culture, history, and movies; Bhavesh, for being the resident Michelin chef and the amazing discussions cum banter; Ranvir, Deva and Jyoti, for their chill attitude and calm demeanor at all times; Deepika, for introducing a variety of excellent TED talks, articles, and perspectives about life; Mona, for being an amazing movie compatriot and for the interesting discussions ranging from tensors to TENET; Hyeji, Shaileshh and Shripad, for their sagely advice at many crucial moments of my grad school; Amish, Saboo, Sayantani and Vegnesh, for being a cheerful presence at all times and for their eclectic interests on a wide range of topics; Vinay Iyer, for being an early example (right from day one of grad school) of how to balance life and work; Weihao, for being always there to discuss about anything from research to sports. I would also like thank my latest group-mates and co, Milind, Ashwin, Viraj, Anant, and Ushasi for the fun company and the wonderful collaborations.

Next, I would like to thank my roommates and close friends, Konik and Sameer, for their amazing camaraderie all these years. I have learnt a lot of valuable things living with them; Konik, for his fearless attitude in approaching problems and a clear cut vision, and Sameer, for his straightforward and gentle approach to various things in life. I would also like to thank the special residents of CSL 104, Harsh and Siddhartha, for always being available to chat about anything under the sun at any time of the day. Harsh, with his encyclopedic knowledge about everything and Siddhartha, with his diverse set of talents ranging from ballroom dancing to improv, have left an indelible mark on me. Special thanks to Unnat for being an amazing friend and a constant source of inspiration. I have learnt a great deal just watching him figure out and follow the paths/strategies to be a world-class researcher. And a big thank you to Pratik, Surya, and Vaishnavi, for being a constant bedrock of support across the years. If I begin writing why and what I would like to thank them for, I am afraid I will run out of space. So I will simply say: thank you for everything!

A heartfelt thanks to Brenda and Peggy for all the administrative help and their cheer-radiating presence in CSL!

Outside the UIUC universe, I would like to thank some of my closest friends from (school, IITB) days who were always there to support me: Vamisdhar, Tarun, Sasank, Soham, Sukanya, Nithin, Sai Kumar, Sainath, Sai Bhargav, Vishal, and Srikar sir. I would also like to thank Aditi Udupa for being an ideal role model, especially at some of the formative years of my life. I am grateful to Sankeerth and Krishnamoorthy Iyer for their timely help with grad school applications.

As with everything in life, my grad school experience has been a rollercoaster ride of emotions. In particular, I went through several health crises in the form of depression and vitamin deficiencies. In those dark moments, if not for the timely presence and support of amazing friends and family, I would not have made it through. A special such thanks to my counselor Emily, who has literally shined the light in my life when I had no hope left. No words can express my gratitude to her. Similarly I would like to think Priya Soundararajan, for being a supportive figure during some of the darkest phases of depression, and for being a wonderful friend putting up with my endless curiosity to talk about anything related to maths, history, and movies. I would also like to thank Vamsi and Ankita without whose presence and support I wouldn't have made it through the latter crisis. My deepest heartfelt thanks to you both!

A special thanks to Pranathi, Murthy, Samanvitha, and Dhruv for making me feel at home being so far from the home!

Finally, in the tradition of saving the best for the last, I would like to express deepest gratitude for my parents. I am eternally thankful to them for their selfless love, sacrifice, support, and encouragement all these years. Without them, none of this would have been possible. To them, I dedicate this thesis.

Table of Contents

List of Tables x

List of Figures xi

Chapter 1 **Introduction** 1
- 1.1 **Organization of this dissertation** 6
- 1.2 **Bibliographical note** 6

Chapter 2 **Mixture-of-Experts: Consistent and Efficient Algorithms** 7
- 2.1 Background on Mixture-of-Experts (MoE) 7
- 2.2 Algorithms 10
- 2.3 The proposed algorithm for learning MoE 11
- 2.4 Theoretical analysis 13
- 2.5 Experiments 17

Chapter 3 **Mixture-of-Experts: Learning via Gradient Descent** 20
- 3.1 Connection between k-MoE and other popular models 20
- 3.2 Optimization landscape design for MoE 21
- 3.3 Experiments 28

Chapter 4 **KO Codes: Novel Neural Codes** 31
- 4.1 Introduction 31
- 4.2 Problem formulation and background 35
- 4.3 KO codes: Novel Neural codes 40
- 4.4 Main results 41
- 4.5 KO codes improve upon Polar codes 48
- 4.6 Tiny KO 50

Chapter 5 **Conclusion** 52

References 54

Appendix A **Appendix for Chapter 2** 61
A.1 Toolbox for method of moments 61
A.2 Class of non-linearities 63
A.3 Proofs of Section 2.4 65
A.4 Proof of Theorem 1 for $k = 2$ 65
A.5 Proof of Theorem 1 for general k 68
A.6 Proof of Theorem 2 69
A.7 Proof of Lemma 8 70
A.8 Proof of Lemma 7 76
A.9 Proof for $k = 2$ 76
A.10 Proof for general k 81
A.11 Gradient EM algorithm 86
A.12 Additional experiments 88

Appendix B **Appendix for Chapter 3** 92
B.1 Valid class of non-linearities 92
B.2 Proofs of Section 3.2.1 94
B.3 Proofs of Section 3.2.2 97
B.4 Additional experiments 106

Appendix C **Appendix for Chapter 4** 107
C.1 Polar$(64, 7)$ code 107
C.2 Gains with list decoding 109
C.3 Discussion 109
C.4 Plotkin construction 110
C.5 KO$(8, 2)$: Architecture and training 110
C.6 Soft-MAP decoder 114
C.7 Experimental details 116
C.8 Results for Order-1 codes 119

List of Tables

2.1 Fit of our learned parameters for non-Gaussian inputs 19
2.2 Performance of our algorithm under orthogonal and non-orthogonal settings 19

4.1 Parameters of a RM(m, r) code 37
4.2 The smaller TinyKO neural architecture with 100 times smaller number of parameters achieve similar bit-error-rates as the bigger KO architecture. 50

List of Figures

1.1 Model for point-to-point communication. 3

1.2 Neural Augmentation on top of existing optimal codes. 4

1.3 Mixture-of-Experts model. 5

2.1 Architecture for k-MoE 8

2.2 Algorithm to learn the MoE parameters. **Algorithm** 1: First we take non-linear transformations on the samples $(\boldsymbol{x}_i, y_i)$ to compute the tensors $\mathcal{T}_2, \mathcal{T}_3$. Spectral decomposition on $\mathcal{T}_2, \mathcal{T}_3$ recovers the regressors. **Algorithm** 2: EM uses the learnt regressors and samples to learn the gating parameters with random initializations 10

2.3 (a): GatingFit for our algorithm under non-orthogonality setting. (b),(c): Estimation error $\mathcal{E}(\boldsymbol{A}, \boldsymbol{W})$ of our algorithm vs. joint-EM algorithm. Our algorithm is significantly better than the joint-EM under random initializations. 18

3.1 Our proposed losses L_4 (defined in Equation 3.2) and L_{log} (defined in Equation 3.4) to learn the respective regressor and gating parameters of a MoE model in Equation 2.1 achieve much better empirical results than the standard methods. 21

3.2 (a), (b): Robustness to parameter orthogonality: Plots show performance over 5 different trials for our losses L_4 and L_{log} respectively. (c) Robustness to Gaussianity of input: Performance over various mixing probabilities p. . . . 28

4.1 KO$(9, 2)$, discovered by training a neural network with a carefully chosen architecture in Section 4.3, significantly improves upon state-of-the-art RM$(9, 2)$ both in BER and BLER. (For both codes, the code block length is $2^9 = 512$ and the number of transmitted message bits is $\binom{9}{0} + \binom{9}{1} + \binom{9}{2} = 55$. Also, both codes are decoded using successive cancellation decoding with similar decoding complexity) . . . 33

4.2 Histogram of pairwise distances between codewords of the KO$(9, 2)$ code shows a strong resemblance to that of the Gaussian codebook, unlike the classical Reed-Muller code RM$(9, 2)$. 34
4.3 Plotkin trees for RM$(3, 1)$ and KO$(3, 1)$ codes. Leaves are shown in green. Red arrows indicate the bit decoding order. . 38
4.4 Neural network based KO$(8, 2)$ and KO-b$(8, 2)$ improve upon RM$(8, 2)$ in BER and BLER, but the gain is small for the binarized codewords of KO-b$(8, 2)$ (for all the codes, the code dimension is 37 and block length is 256). 42
4.5 Histograms of pairwise distances between codewords for $(8, 2)$ codes reveal that KO$(8, 2)$ code has learned an approximate Gaussian codebook that can be efficiently decoded. 44
4.6 Separating each sub-code contribution in the KO$(8, 2)$ decoder and the RM$(8, 2)$ decoder reveals that KO$(8, 2)$ improves in the total BLER by balancing the contributions more evenly over the sub-codes. 45
4.7 KO$(8, 2)$ trained on AWGN is robust when tested on a fast fading channel and maintains a significant gain over RM(8,2). 46
4.8 KO$(8, 2)$ trained on AWGN is robust when tested on a bursty channel and maintains a significant gain over RM$(8, 2)$. 46
4.9 Ablation studies highlight that both non-linearity and real-valued codewords are equally important for good performance of KO codes. The linear version, KO-linear$(8, 2)$, and the binary version, KO-b$(8, 2)$, both perform worse than KO$(8, 2)$ and similar to RM$(8, 2)$. 47
4.10 Neural network based KO code improves upon the Polar$(64, 7)$ code when trained on AWGN channel. KO decoder also matches the optimal MAP decoder. 49

A.1 Plot of parameter estimation error with varying number of samples(n): (a) $n = 1000$ (b) $n = 5000$. (c) $n = 10000$. 88
A.2 Parameter estimation error for the sigmoid and ReLU non-linearities respectively. 89
A.3 Prediction error for the concrete, stock portfolio and the airfoil data sets respectively. 89

B.1 Comparison of SGD on our losses $(L_4, L_{\log})$ vs. ℓ_2 and the EM algorithm. 106

C.2 KO code achieves a significant gain over the Polar$(64, 7)$ code in BLER when trained on AWGN channel. KO decoder also matches the optimal MAP decoder. 108

C.6 KO$(6,1)$ code. **Left**: KO$(6,1)$ code achieves significant gain over RM$(6,1)$ code (with Dumer) when trained on AWGN channel. **Right**: Under the optimal MAP decoding, KO$(6,1)$ and RM$(6,1)$ codes achieve the same performance. Error rates for a random Gaussian codebook are also plotted as a baseline. 120
C.1 Plotkin trees for the Polar$(64,7)$ encoder and our neural KO encoder counterpart. Both codes have dimension $k = 7$ and blocklength $n = 64$. 121
C.3 The same KO(8,2) encoder and decoder as those used in Figure 4.4 achieve a significant gain (without any retraining or fine-tuning) when list decoding is used together with the KO decoder. The magnitude of the gain is comparable to the gain achieved by the same list decoding technique on the successive cancellation decoder of the RM(8,2) code. We used the list decoding from [1] but without the permutation technique. 122
C.4 Plotkin trees for RM(8,2) and KO(8,2) encoders. Leaves are highlighted in green. Both codes have dimension $k = 37$ and blocklength $n = 256$. 123
C.5 Plotkin trees for the RM$(8,2)$ and KO$(8,2)$ decoders. Red arrows indicate the bit decoding order. 124

Chapter 1

Introduction

Undeniably, thanks to modern digital communication, we all live in the age of information. No matter where we are located, we are able to communicate and transmit messages at ever increasing speeds. From the age of messenger pigeons to that of electronic email, how did we reach here? What makes the modern communication systems so reliable and effective?

Codes, composed of encoder and decoder pairs, are the basic mathematical objects that enable reliable communication in the presence of noise: encoder maps original data bits into a longer sequence, and decoders map the received noisy sequence to the original bits. Through his ground-breaking work in [2], the birth of information theory, Shannon laid the foundations for modern digital communication by providing a mathematical lens to study and understand codes. Ever since, much progress has been made in the design of optimal codes that can be reliably transmitted and efficiently decoded under noisy conditions. This is the primary focus of *coding theory*. Landmark codes include Reed-Muller (RM) codes, Bose–Chaudhuri–Hocquenghem (BCH) codes, convolutional codes, turbo codes, low density parity check (LDPC) codes and polar codes [3]; each is a linear code and represents a mathematical breakthrough. Their impact on humanity is huge: each of these codes has been used in global communication standards over the past six decades. Put more simply, every cellular phone designed ever uses one of these codes.

Given the practical relevance, inventing codes is a major intellectual activity in both academia and the wireless industry. On the other hand, designing codes is a challenging task, mostly driven by human ingenuity. Befittingly, the progress has been sporadic historically. For example, the time duration between convolutional codes (2^{nd} generation cellular standards) to polar codes (5^{th} generation cellular standards) is over four decades. The core challenge is that the space of codes is very vast and the sizes astronomical; for instance a rate 1/2 code over even 100 information bits involves designing 2^{100} codewords in a 200 dimensional space. Hence computationally efficient encoding and decoding procedures are a must, apart from high reliability. Thus, although a random code is information theoretically optimal, neither encoding nor decoding is computationally efficient. However, the mathematical landscape of computationally efficient codes has been plumbed over the decades by some of the finest mathematical minds, resulting in two distinct families of codes: *algebraic codes* (RM, BCH – focused on properties of polynomials) and *graph codes* (Turbo, LDPC – based on sparse graphs and statistical physics). The former is deterministic and involves discrete mathematics, while the latter harnesses randomness, graphs, and statistical physics to behave like a pseudorandom code. A major open question in coding theory is the invention of new codes and especially fascinating would be a family of codes outside of these two classes.

On the other hand, in recent years we have seen tremendous success of deep learning (DL) across a variety of disciplines such as computer vision [4], natural language processing [5], and game playing [6, 7]. A key hallmark of these results is that they have transformed several domains of human endeavor that have traditionally relied on mathematical ingenuity, e.g., game playing (AlphaGo, AlphaZero). Especially in the context of game playing, where the game model and the objective are mathematically well defined, DL has shown the ability to learn complex *algorithms/strategies* that outperform humans purely from data alone. Inspired by these successes, we are thus motivated to ask:

Question 1. *Can we harness the potential of deep learning to address the long standing goals of coding theory?*

Addressing this question is the primary focus of this thesis.

To this end, let's first recall the mathematical model for point-to-point communication [2] as illustrated in Figure 1.1: A message $\boldsymbol{m} \in \{1, 2, \ldots, M\}$ that we wish to transmit is randomly generated. It is further processed by an *encoder* which maps the message to a corresponding codeword that is transmitted across a noisy channel medium. We consider the additive white Gaussian noise (AWGN) channel; for a variety of theoretical and practical reasons, AWGN channel is the canonical setting to benchmark codes. Upon receiving the corrupted codeword, the *decoder* estimates the message as $\hat{\boldsymbol{m}}$. Ideally, we desire $\hat{\boldsymbol{m}}$ to be the same as $\boldsymbol{m}$. Thus, the defining criterion for

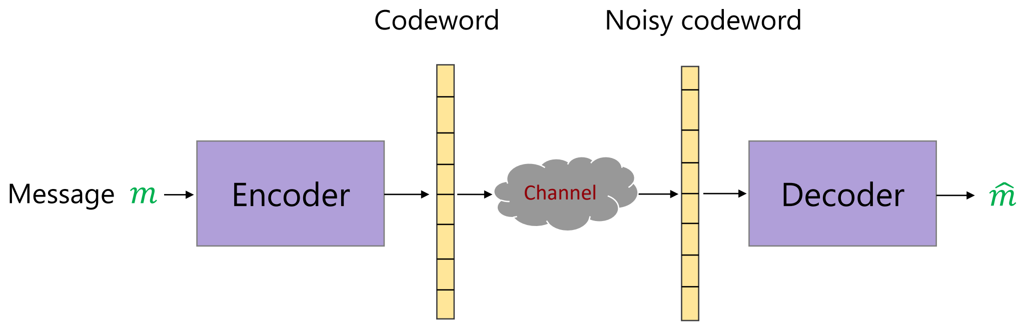


Figure 1.1: Model for point-to-point communication.

any (encoder, decoder) pair is its probability of block error: $\mathbb{P}\left(\hat{\boldsymbol{m}} \neq \boldsymbol{m}\right)$, the smaller the probability better the reliability of the code. The primary goal of channel coding is to design codes that achieve state-of-the-art reliability performance over the AWGN channel.

Recall that the classical optimal codes such as turbo, LDPC, and polar all have *linear* encoders and are *binary* valued. In order to design a much broader class of *non-linear* and *real* valued codes, a natural idea is to appropriately parameterize the (encoding, decoding) blocks in Figure 1.1 with standard neural networks (NNs). To train these NNs, we can use an end-to-end cross entropy loss function which serves as a differentiable surrogate to the error probability metric. However, this alone does not suffice to learn good codes that achieve near state-of-the-art results. Indeed, as highlighted in the works of [8, 9], in the absence of any structure, NNs fail to learn non-trivial codes and end up performing worse than a simple repetition code (repeating each message bit multiple times). A fundamental question in machine learning for channel coding is thus: how do we design architectures for our neural encoders and decoders that give the appropriate inductive bias?

To this end, we rely on the principle of *neural augmentation* [10]: cleverly augment the existing encoders and decoders with their corresponding neural counter parts. The underlying intuition behind this is quite simple, harness the best of both worlds: the human engineered codes and their neural versions. In other words, we capitalize on the existing optimal codes and use them as an inductive bias to design new neural codes. Hence we can expect that the codes learnt via this approach are strictly better than their non-neural

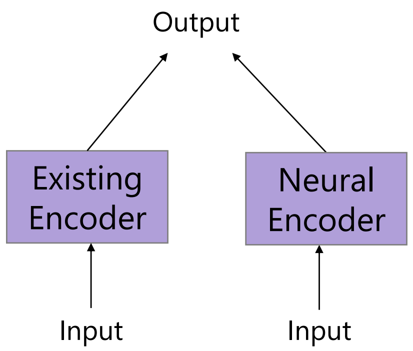


Figure 1.2: Neural Augmentation on top of existing optimal codes.

counterparts. This abstract schematic is illustrated in Figure 1.2 in the context of a designing a new encoder (similarly for decoder).

In view of this, it is thus natural to ask what are some canonical models that realize this abstraction. Indeed, a well-known model called *Mixture-of-Experts (MoE)* fits the task. The canonical model for two Mixture-of-Experts is illustrated in Figure 1.3. Here each of the experts represent a function block that map the input to its corresponding output. Of significance is the presence of a gating mechanism which influences how the individual expert's decisions are modulated into obtaining a final output. Note that the gating mechanism is input *dependent*. Hence the gating automatically learns when/how to factor in the best expert's decision. We note that the MoE model is more general than the neural augmented approach since MoE allows for learnable components in both the experts and the gating mechanism. Also MoE is a rich mathematical model in its own right and has received a lot of theoretical and empirical interest [11–18] ever since its inception more than two decades ago [19].

However, belying its empirical success, very little is known about the canonical MoE model with regards to learning its parameters. In Chapter 2, we study the MoE in great detail and address this gap. We provide the first efficient and consistent algorithm with global learning guarantees for learning the paramaters in a MoE which has been an open problem for more than two decades. Our algorithm uses a novel combination of spectral methods and the classical EM algorithm which are of independent mathematical interest.

In Chapter 3, we further improve upon these algorithms and present

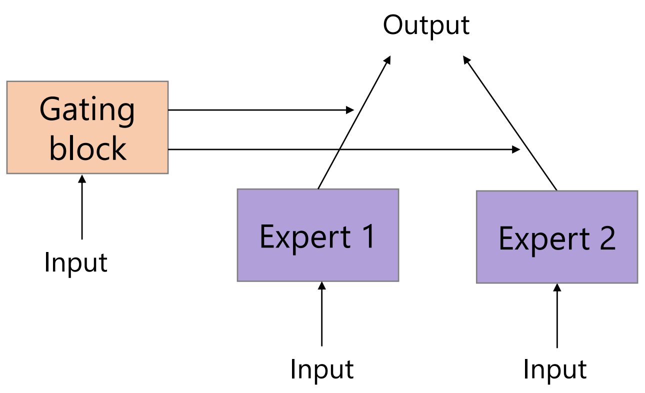


Figure 1.3: Mixture-of-Experts model.

a scalable gradient descent (GD) based approach with an end-to-end loss function to learn the MoE parameters and provide the first finite sample guarantees of any kind for MoE. More precisely, we construct two non-trivial non-convex loss functions to learn the expert and the gating parameters respectively. Our loss functions possess nice landscape properties such as all local minima being global and the global minima corresponding to the ground truth parameters. We further show that gradient descent on our losses can recover the true parameters with global/random initializations. To the best of our knowledge, ours is the first GD based approach with finite sample guarantees to learn the parameters of MoE.

Chapter 4 pivots from the MoE model and focuses on the design of codes in a data driven manner. Our major result is the invention of *KO codes*, a computationally efficient family of deep-learning driven codes that outperform the state-of-the-art RM and Polar codes, in the challenging short-to-medium block length regime on the AWGN channel. The key technical innovation behind KO codes is the design of a novel family of neural architectures inspired by the computation tree of the **K**ronecker **O**peration (KO) that is central to RM and Polar codes (neural augmentation principle). These architectures pave way for the discovery of a much richer class of nonlinear algebraic structures. Surprisingly, despite being heavily structured, KO codes exhibit random Gaussian like behavior! These results highlight the potential of DL in inventing a rich family of hitherto unexplored non-linear codes.

Together, our results thus answer the central question of this dissertation, Question 1, in affirmative.

1.1 Organization of this dissertation

This dissertation is broadly in two parts: Chapter 2 and Chapter 3 focus on the MoE model and provide efficient and provable algorithms to learn its parameters. Chapter 4 focuses on the paradigm of designing codes in a data-driven manner using deep learning and details the KO codes. Each of these two parts can be read independently of one another. All the necessary preliminaries and the main results are provided within each chapter.

In Chapter 5 we outline some interesting future directions to the topics covered in this dissertation.

1.2 Bibliographical note

The following is a chapter-wise list of the publications that include the work presented herein:

Chapter 2

- A. V. Makkuva, S. Oh, S. Kannan, and P. Viswanath, "Breaking the gridlock in mixture-of-experts: Consistent and efficient algorithms," *Proceedings of the* 36*th* *International Conference on Machine Learning (ICML) 2019.*

Chapter 3

- A. V. Makkuva, S. Oh, S. Kannan, and P. Viswanath, "Learning in gated neural networks," *Proceedings of the* 23*rd* *International Conference on Artificial Intelligence and Statistics (AISTATS) 2020.*

Chapter 4

- A. V. Makkuva*, X. Liu*, M. V. Jamali, H. Mahdavifar, S. Oh, and P. Viswanath, "KO codes: inventing nonlinear encoding and decoding for reliable wireless communication via deep-learning," *Proceedings of the* 38*th* *International Conference on Machine Learning (ICML) 2021.*

Chapter 2

Mixture-of-Experts: Consistent and Efficient Algorithms

2.1 Background on Mixture-of-Experts (MoE)

In this section, we study a popular gated neural network architecture known as Mixture-of-Experts (MoE). MoE is a basic building block of highly successful modern neural networks like Gated Recurrent Units (GRU) and Attention networks. A key interesting feature of MoE is the presence of a gating mechanism that allows for specialization of experts in their respective domains. MoE allows for the underlying expert models to be simple while allowing to capture complex non-linear relations between the data. Ever since their inception more than two decades ago [19], they have been a subject of great research interest [11–18] across multiple domains such as computer vision, natural language processing, speech recognition, finance, and forecasting.

The basic MoE model is the following: let $\boldsymbol{x} \in \mathbb{R}^d$ be the input feature vector and $y \in \mathbb{R}$ be the corresponding label. Then the discriminative model $P_{y|\boldsymbol{x}}$ for the k-mixture of experts (k-MoE) in the regression setting is:

$$\begin{aligned} P_{y|\boldsymbol{x}} &= \sum_{i=1}^{k} P_{i|\boldsymbol{x}} P_{y|\boldsymbol{x},i} \\ &= \sum_{i=1}^{k} \frac{e^{\langle \boldsymbol{w}_i^*, \boldsymbol{x} \rangle}}{\sum_j e^{\langle \boldsymbol{w}_j^*, \boldsymbol{x} \rangle}} \mathcal{N}(y | g(\langle \boldsymbol{a}_i^*, \boldsymbol{x} \rangle), \sigma^2). \end{aligned} \tag{2.1}$$

Figure 2.1 details the architecture for k-MoE.

The interpretation behind Equation 2.1 is that for each input $\boldsymbol{x}$, the gating network chooses an expert based on the outcome of a multinomial random variable $z \in [k]$, whose probability depends on $\boldsymbol{x}$ in a parametric way, i.e. $z|\boldsymbol{x} \sim \mathrm{softmax}(\langle \boldsymbol{w}_1^*, \boldsymbol{x} \rangle, \ldots, \langle \boldsymbol{w}_k^*, \boldsymbol{x} \rangle)$. The chosen expert then generates the output y from a Gaussian distribution centred at a non-linear activation of

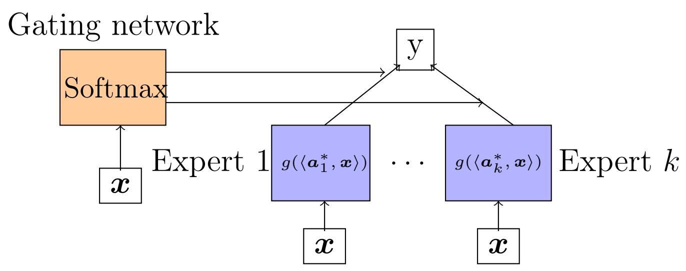

Gating network
Softmax
y
$\boldsymbol{x}$
Expert 1
$g(\langle \boldsymbol{a}_1^*, \boldsymbol{x}\rangle)$
$\cdots$
$g(\langle \boldsymbol{a}_k^*, \boldsymbol{x}\rangle)$
Expert k
$\boldsymbol{x}$
$\boldsymbol{x}$

Figure 2.1: Architecture for k-MoE

$\boldsymbol{x}$, i.e. $g(\langle \boldsymbol{a}_z^*, \boldsymbol{x}\rangle)$, with variance σ^2. We want to learn the expert parameters $\boldsymbol{a}_i^* \in \mathbb{R}^d$ (also referred to as the regressors) and the gating parameters $\boldsymbol{w}_i^* \in \mathbb{R}^d$, assuming we know the non-linear activation $g : \mathbb{R} \to \mathbb{R}$.

This problem of learning MoE has been a long standing open problem for more than two decades, even though it is a fundamental building block of several state-of-the-art gated neural network architectures. Gated neural networks such as GRUs and Sparsely-gated-MoEs have been widely successful in challenging tasks like machine translation [20–22]. Parameters are typically learnt through (stochastic) gradient descent on a non-convex loss function. However, these methods do not possess any theoretical guarantees, even for the simplest gated neural network, the MoE.

On the other hand, existing guarantees for simpler models without gating units do not extend to MoEs. Consider the mixture of generalized linear models (M-GLMs) [23–26], a strict simplification of the k-MoE model in Equation 2.1, where $\boldsymbol{w}_i^* = 0$ for all $i \in \{1, \ldots, k\}$. The learning in M-GLMs is usually done through a combination of spectral methods and greedy methods such as EM. A major limitation of these methods is that they rely critically on the fact that the mixing probability is a constant and hence they do not generalize to MoEs (see Section 2.2). In addition, the EM algorithm, the workhorse for learning in parametric mixture models, is prone to bad local minima [23, 26, 27] (we independently verify this for MoEs in Section 2.5). These theoretical shortcomings and practical relevance of the MoE models lead to the following fundamental question:

Can we find an efficient and a consistent algorithm (with global initializations) that recovers the true parameters of the model with theoretical guarantees?

In this chapter, we address this question precisely and make the following contributions:

1) First theoretical guarantees: We provide the first (poly-time) efficient algorithm that recovers the true parameters of a MoE model with global initializations (Theorem 1 and Theorem 2). We allow for a wide class of non-linearities which includes the popular choices of identity, sigmoid, and ReLU. To the best of our knowledge, ours is the first work to give global convergence guarantees for MoE.

2) Algorithmic innovations: Existing algorithms jointly or iteratively estimate the expert parameters and the gating paramters in the MoE and can get stuck in local minima. In this section, we propose a novel algorithm that breaks the gridlock and can directly estimate the expert parameters by sensing its echo in a cross-moment tensor between the inputs and the output (Algorithm 1 and Algorithm 2). Once the experts are known, the recovery of gating parameters still requires an EM algorithm; however, we show that the EM algorithm for this simplified problem, unlike the joint EM algorithm, converges to the true parameters. The proofs of global convergence of EM as well as the design of the cross-moment tensor are of independent mathematical interest.

3) Novel transformations: In this section, we introduce the novel notion of "Cubic and Quadratic Transform (CQT)". These are polynomial transformations on the output labels tailored to specific non-linear activation functions and the noise variance. The key utility of these transforms is to equip MoEs with a supersymmetric tensor structure in a principled way (Theorem 1).

Notation. In this section, we denote Euclidean vectors by bold face lowercase letters $\boldsymbol{a}, \boldsymbol{b}$, etc., and scalars by plain lowercase letters y, z, etc. We use $\mathcal{N}(y|\mu, \sigma^2)$ either to denote the density or the distribution of a Gaussian random variable y with mean μ and variance σ^2, depending on the context. $[d] \triangleq \{1, \ldots, d\}$. Perm$[d]$ denotes the set of all permutations on $[d]$. We use $\otimes$ to denote the tensor outer product of vectors in $\mathbb{R}^d$. $\boldsymbol{x}^{\otimes 3}$ denotes $\boldsymbol{x} \otimes \boldsymbol{x} \otimes \boldsymbol{x}$, where $(\boldsymbol{x} \otimes \boldsymbol{x} \otimes \boldsymbol{x})_{ijk} = x_i x_j x_k$. $\mathrm{sym}(\boldsymbol{x} \otimes \boldsymbol{y} \otimes \boldsymbol{z})$ denotes the symmetrized version of $\boldsymbol{x} \otimes \boldsymbol{y} \otimes \boldsymbol{z}$, i.e. $\mathrm{sym}(\boldsymbol{x} \otimes \boldsymbol{y} \otimes \boldsymbol{z})_{ijk} = \sum_{\sigma \in \mathrm{Perm}[d]} x_{\sigma(i)} y_{\sigma(j)} z_{\sigma(k)}$. $\boldsymbol{e}_i, i \in [d]$ denotes the standard basis vectors for $\mathbb{R}^d$. Through out, we assume that $\boldsymbol{w}_k^* = 0$, without loss of generality.

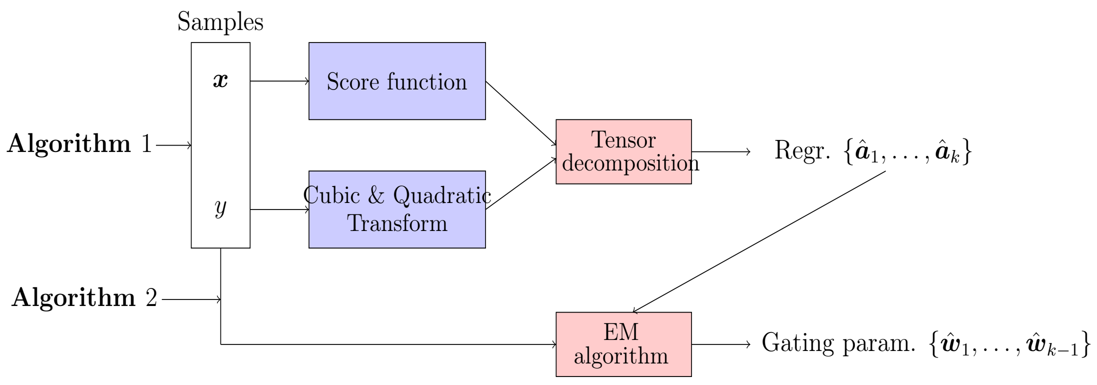

Samples
$\boldsymbol{x}$
y
Algorithm 1
Algorithm 2
Score function
Cubic & Quadratic Transform
Tensor decomposition
Regr. $\{\hat{\boldsymbol{a}}_1, \ldots, \hat{\boldsymbol{a}}_k\}$
EM algorithm
Gating param. $\{\hat{\boldsymbol{w}}_1, \ldots, \hat{\boldsymbol{w}}_{k-1}\}$

Figure 2.2: Algorithm to learn the MoE parameters. **Algorithm** 1: First we take non-linear transformations on the samples $(\boldsymbol{x}_i, y_i)$ to compute the tensors $\mathcal{T}_2, \mathcal{T}_3$. Spectral decomposition on $\mathcal{T}_2, \mathcal{T}_3$ recovers the regressors. **Algorithm** 2: EM uses the learnt regressors and samples to learn the gating parameters with random initializations

2.2 Algorithms

In this section, we present our algorithms to learn the regression and gating parameters *separately*. Figure 2.2 summarizes our algorithm. First we take a moment to highlight the issues of the existing approaches.

For illustration purposes, we suppose that $k = 2$ in Equation 2.1. We assume without loss of generality that $\boldsymbol{w}_k^* = \boldsymbol{w}_2^* = 0$ and denote $\boldsymbol{w}_1^* = \boldsymbol{w}^*$. Thus the 2-MoE model is given by $P_{y|\boldsymbol{x}}$:

$$\frac{e^{\langle \boldsymbol{w}^*, \boldsymbol{x}\rangle} \mathcal{N}(y|g(\langle \boldsymbol{a}_1^*, \boldsymbol{x}\rangle), \sigma^2)}{1 + e^{\langle \boldsymbol{w}^*, \boldsymbol{x}\rangle}} + \frac{\mathcal{N}(y|g(\langle \boldsymbol{a}_2^*, \boldsymbol{x}\rangle), \sigma^2)}{1 + e^{\langle \boldsymbol{w}^*, \boldsymbol{x}\rangle}} \tag{2.2}$$

Issues with traditional tensor methods. In the far simplified setting of the absence of the gating parameter, i.e. $\boldsymbol{w}^* = 0 \in \mathbb{R}^d$, we see that 2-MoE reduces to 2-uniform mixture of GLMs. In this case, for $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$, the standard approach is to construct a 3rd-order tensor $\mathcal{T}$ by regressing the output y on the score transformation $\mathcal{S}_3(\boldsymbol{x}) \triangleq \boldsymbol{x} \otimes \boldsymbol{x} \otimes \boldsymbol{x} - \sum_{i \in [d]} \mathrm{sym}\left(\boldsymbol{x} \otimes \boldsymbol{e}_i \otimes \boldsymbol{e}_i\right)$, i.e.

$$\begin{aligned} \mathcal{T} \triangleq \mathbb{E}[y \cdot \mathcal{S}_3(\boldsymbol{x})] = \frac{1}{2}\mathbb{E}[g'''(\langle \boldsymbol{a}_1^*, \boldsymbol{x}\rangle)] \cdot (\boldsymbol{a}_1^*)^{\otimes 3} \\ + \frac{1}{2}\mathbb{E}[g'''(\langle \boldsymbol{a}_2^*, \boldsymbol{x}\rangle)] \cdot (\boldsymbol{a}_2^*)^{\otimes 3} \, . \end{aligned} \tag{2.3}$$

Here the second equality follows from the generalized Stein's lemma that

$\mathbb{E}[f(\boldsymbol{x}) \cdot \mathcal{S}_3(\boldsymbol{x})] = \mathbb{E}[\nabla_{\boldsymbol{x}}^{(3)} f(\boldsymbol{x})]$ under some regularity conditions on $f : \mathbb{R}^d \mapsto \mathbb{R}$. Then the regressors can be learned through spectral decomposition on $\mathcal{T}$, where the uniqueness of decomposition follows from [28]. If we apply a similar technique for 2-MoE in Equation 2.2, we obtain that

$$\begin{aligned} \mathbb{E}[y \cdot \mathcal{S}_3(\boldsymbol{x})] = \sum_{i=1,2} \alpha_i (\boldsymbol{a}_i^*)^{\otimes 3} + \beta_i \operatorname{sym}(\boldsymbol{a}_i^* \otimes \boldsymbol{a}_i^* \otimes \boldsymbol{w}^*) \\ + \gamma_i \operatorname{sym}(\boldsymbol{a}_i^* \otimes \boldsymbol{w}^* \otimes \boldsymbol{w}^*) + \delta(\boldsymbol{w}^*)^{\otimes 3}, \end{aligned} \tag{2.4}$$

where $\alpha_i, \beta_i, \gamma_i, \delta$ are some scalar constants depending on the parameters $\boldsymbol{a}_1^*, \boldsymbol{a}_2^*, \boldsymbol{w}^*$ and g. Thus Equation 2.4 reveals that traditional spectral methods do not yield a supersymmetric tensor of the desired parameters for MoEs. In fact, Equation 2.4 contains all the 3rd-order rank-1 terms formed by $\boldsymbol{a}_1^*, \boldsymbol{a}_2^*$ and $\boldsymbol{w}^*$. Hence we cannot recover these parameters uniquely. Note that the inherent coupling between the regressors $\boldsymbol{a}_1^*, \boldsymbol{a}_2^*$ and the gating parameter $\boldsymbol{w}^*$ in Equation 2.2 manifests as a cross tensor in Equation 2.4. This coupling serves as a key limitation for the traditional methods which critically rely on the fact that the mixing probability $p = \frac{1}{2}$ in Equation 2.4 is a constant. In fact, we recover Equation 2.3 by letting $\boldsymbol{w}^* = 0$ in Equation 2.4.

Issues with EM algorithm. EM algorithm is the workhorse for parameter learning in both the k-MoE and HME models [29]. However, it is well known that EM is prone to spurious minima and existing theoretical results only establish local convergence for the regressors and the gating parameters. Indeed, our numerical experiments in Section 2.5.3 verify this fact. Figure 2.3b and Figure 2.3c highlight that joint-EM often gets stuck in bad local minima.

2.3 The proposed algorithm for learning MoE

In order to tackle these challenges, we take a different route and propose to estimate the regressors and gating parameters *separately*. To gain intuition about our approach, let us consider 2-MoE model in Equation 2.2 with $\sigma = 0$ and linear g. Then we have that y either equals $\langle \boldsymbol{a}_1^*, \boldsymbol{x} \rangle$ with probability $\sigma(\langle \boldsymbol{w}^*, \boldsymbol{x} \rangle)$ or equals $\langle \boldsymbol{a}_2^*, \boldsymbol{x} \rangle$ with probability $1 - \sigma(\langle \boldsymbol{w}^*, \boldsymbol{x} \rangle)$, where $\sigma(\cdot)$ is the sigmoid function. If we exactly know $\boldsymbol{w}^*$, we can recover $\boldsymbol{a}_1^*$ and $\boldsymbol{a}_2^*$ by solving a simple linear regression problem since we can recover the true latent variable $\boldsymbol{z} \in \{1, 2\}$ with high probability. Similarly, if we know $\boldsymbol{a}_1^*$ and $\boldsymbol{a}_2^*$, it is easy to see that we can recover $\boldsymbol{w}^*$ by solving a binary linear classification problem. Thus knowing either the regressors or the gating parameters makes the estimation of other parameters easier. However, how do we first obtain one set of parameters without any knowledge about the other?

Our approach precisely addresses this question and breaks the *grid lock*. We show that we can extract the regressors $\boldsymbol{a}_1^*$ and $\boldsymbol{a}_2^*$ without knowing $\boldsymbol{w}^*$ at all, just using the samples. Although we explain our approach with two mixtures, all claims are made precise for general k in Theorems 1 and 2, and the algorithms are written for general k as well in Algorithms 1 and 2.

Step 1: Estimation of regressors

To learn the regressors, we first pre-process $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$ using the score transformations $\mathcal{S}_3$ and $\mathcal{S}_2$, i.e.

$$\mathcal{S}_3(\boldsymbol{x}) \triangleq \boldsymbol{x} \otimes \boldsymbol{x} \otimes \boldsymbol{x} - \sum_{i \in [d]} \mathrm{sym}\left(\boldsymbol{x} \otimes \boldsymbol{e}_i \otimes \boldsymbol{e}_i\right), \tag{2.5}$$

$$\mathcal{S}_2(\boldsymbol{x}) \triangleq \boldsymbol{x} \otimes \boldsymbol{x} - I. \tag{2.6}$$

These score functions can be viewed as higher-order feature extractors from the inputs. As we have seen in Equation 2.3, these transformations suffice to learn the parameters in M-GLMs. However this approach fails in the context of MoE, as highlighted in Equation 2.4. Can we still construct a supersymmetric tensor for MoE?

To answer this question in a principled way, we introduce the notion of "Cubic and Quadratic Transform (CQT)" for the labels, i.e.

$$\mathcal{P}_3(y) \triangleq y^3 + \alpha y^2 + \beta y, \quad \mathcal{P}_2(y) \triangleq y^2 + \gamma y.$$

The coefficients (α, β, γ) in these polynomial transforms are obtained by solving a linear system of equations. For the special case of $g =$linear, we obtain $\mathcal{P}_3(y) = y^3 - 3(1+\sigma^2)\, y$ and $\mathcal{P}_2(y) = y^2$. These special transformations are specific to the choice of non-linearity g and the noise variance σ. The key intuition behind the design of these transforms is that we can nullify the cross moments and obtain supersymmetric tensor in Equation 2.3 if we regress

$\mathcal{P}_3(y)$ instead of y, for properly chosen constants α and β. This is made mathematically precise in Theorem 1. A similar argument holds for $\mathcal{P}_2(y)$ too. In addition, the choice of these polynomials is unique in the sense that any other polynomial transformations fail to yield the desired tensor structure. Using these transforms, we construct two special tensors $\hat{\mathcal{T}}_3 \in (\mathbb{R}^d)^{\otimes 3}$ and $\hat{\mathcal{T}}_2 \in (\mathbb{R}^d)^{\otimes 2}$. Later we use the robust tensor power method [30] on these tensors to learn the regressors. Algorithm 1 details our learning procedure. Theorem 1 establishes the theoretical justification for our algorithm.

Algorithm 1 Learning the regressors

```
1: Input: Samples $(\boldsymbol{x}_i, y_i), i \in [n]$
2: Compute $\hat{\mathcal{T}}_3 = \frac{1}{n}\sum_{i=1}^{n} \mathcal{P}_3(y_i) \cdot \mathcal{S}_3(\boldsymbol{x}_i)$ and $\hat{\mathcal{T}}_2 = \frac{1}{n}\sum_{i=1}^{n} \mathcal{P}_2(y_i) \cdot \mathcal{S}_2(\boldsymbol{x}_i)$
3: $\hat{\boldsymbol{a}}_1, \ldots, \hat{\boldsymbol{a}}_k$ = Rank-$k$ tensor decomposition on $\hat{\mathcal{T}}_3$ using $\hat{\mathcal{T}}_2$
```

Step 2: Estimation of gating parameters

To gain intuition for estimating the gating parameters, recall that the traditional joint-EM algorithm randomly initializes both the regressors and the gating parameters and updates them iteratively. Figure 2.3b and Figure 2.3c highlight that this procedure is prone to spurious minima. Can we still learn the gating parameters with *global initializations*? To address this question, we utilize the regressors learnt from Algorithm 1. In particular, we use EM algorithm to update *only* the gating parameters, while fixing the regressors $\hat{\boldsymbol{a}}_1, \ldots, \hat{\boldsymbol{a}}_k$. We show in Theorem 2 that, with *global/random* initializations, this variant of EM algorithm learns the true parameters. To the best of our knowledge, this is the first global convergence result for EM for $k > 2$ mixtures. This motivates the following algorithm ($\varepsilon > 0$ is some error tolerance):

2.4 Theoretical analysis

In this section, we provide the theoretical guarantees for our algorithms in the population setting. We first formally state our assumptions and justify the rationale behind them:

1. $\boldsymbol{x}$ follows standard Gaussian distribution, i.e. $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$.

Algorithm 2 Learning the gating parameter

```
1: Input: Samples $(\boldsymbol{x}_i, y_i), i \in [n]$ and regressors $\hat{\boldsymbol{a}}_1, \ldots, \hat{\boldsymbol{a}}_k$ from Algorithm 1
2: $t \leftarrow 0$
3: Initialize $\boldsymbol{w}_0$ uniformly randomly in its domain $\Omega$
4: while (Estimation error $< \varepsilon$ ) do
5:     Compute the posterior $p^{(i)}_{\boldsymbol{w}_t}$ according to Equation 2.8 for each $j \in [k]$ and $i \in [n]$
6:     Compute $Q(\boldsymbol{w}|\boldsymbol{w}_t)$ according to Equation 2.7 using empirical expectation
7:     Set $\boldsymbol{w}_{t+1} = \text{argmax}_{\boldsymbol{w} \in \Omega} Q(\boldsymbol{w}|\boldsymbol{w}_t)$
8:     $t \leftarrow t + 1$
9:     Estimation error $= \|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\|$
10: end while
```

2. $\|\boldsymbol{a}_i^*\|_2 = 1$ for $i \in [k]$ and $\|\boldsymbol{w}_i^*\|_2 \leq R$ for $i \in [k-1]$, with some $R > 0$.

3. $\boldsymbol{a}_i^*, i \in [k]$ are linearly independent and $\boldsymbol{w}_i^*$ is orthogonal to $\text{span}\{\boldsymbol{a}_1^*, \ldots, \boldsymbol{a}_k^*\}$ for $i \in [k-1]$.

4. The non-linearity $g : \mathbb{R} \to \mathbb{R}$ is (α, β, γ)-valid. For example, this class includes g =linear, sigmoid and ReLU.

Remark. We note that the Gaussianity of the input distribution and norm constraints on the parameters are standard assumptions in the learning of neural networks literature [31–36] and also that of M-GLMs [23, 25–27]. An interpretation behind Assumption 3 is that if we think of $\boldsymbol{x}$ as a high-dimensional feature vector, distinct sub-features of $\boldsymbol{x}$ are used to perform the two distinct tasks of classification (using $\boldsymbol{w}_i^*$'s) and regression (using $\boldsymbol{a}_i^*$'s). We note that we need the above assumptions only for the technical analysis. In Section 2.5.1 and Section 2.5.2, we empirically verify that our algorithms work well in practice even under the relaxation of these assumptions. Thus we believe that the assumptions are merely technical artifacts.

We are now ready to state our results.

Theorem 1 (Recovery of regression parameters)**.** *Let $(\boldsymbol{x}, y)$ be generated according the true model Equation 2.1. Under the above assumptions, we have* *that*

$$\mathcal{T}_2 \triangleq \mathbb{E}[\mathcal{P}_2(y)\cdot\mathcal{S}_2(\boldsymbol{x})] = \sum_{i=1}^{k} c'_g \mathbb{E}[P_{i|\boldsymbol{x}}]\cdot \boldsymbol{a}_i^* \otimes \boldsymbol{a}_i^*,$$
$$\mathcal{T}_3 \triangleq \mathbb{E}[\mathcal{P}_3(y)\cdot\mathcal{S}_3(\boldsymbol{x})] = \sum_{i=1}^{k} c_{g,\sigma} \mathbb{E}[P_{i|\boldsymbol{x}}]\cdot \boldsymbol{a}_i^* \otimes \boldsymbol{a}_i^* \otimes \boldsymbol{a}_i^*,$$

where c'_g and $c_{g,\sigma}$ are two non-zero constants depending on g and σ. Hence the regressors $\boldsymbol{a}_i^*$'s can be learnt through tensor decomposition on $\mathcal{T}_2$ and $\mathcal{T}_3$.

Once we obtain $\mathcal{T}_2$ and $\mathcal{T}_3$, the recovery gurantees for the regressors $\boldsymbol{a}_i^*$ follow from the standard tensor decomposition guarantees, for example, Theorem 4.3 and Theorem 5 of [30]. We assume that the learnt regressors $\boldsymbol{a}_i$ are such that $\max_{i\in[k]} \|\boldsymbol{a}_i - \boldsymbol{a}_i^*\|_2 = \sigma^2\varepsilon$ for some $\varepsilon > 0$. Now we present our theoretical results for global convergence of EM. First we briefly recall the algorithm. Let Ω denote the domain of our gating parameters, defined as

$$\Omega = \{\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{k-1}) : \|\boldsymbol{w}_i\|_2 \le R, \forall i \in [k-1]\}.$$

Then the population EM for the mixture of experts consists of the following two steps:

- **E-step:** Using the current estimate $\boldsymbol{w}_t$ to compute the function $Q(\cdot|\boldsymbol{w}_t)$,
- **M-step:** $\boldsymbol{w}_{t+1} = \mathrm{argmax}_{\boldsymbol{w}\in\Omega} Q(\boldsymbol{w}|\boldsymbol{w}_t)$,

where the function $Q(.|\boldsymbol{w}_t)$ is the expected log-likelihood of the complete data distribution with respect to current posterior distribution. Mathematically,

$$\begin{aligned} Q(\boldsymbol{w}|\boldsymbol{w}_t) &\triangleq \mathbb{E}_{(\boldsymbol{x},y)}\mathbb{E}_{P_{z|\boldsymbol{x},y,\boldsymbol{w}_t}}[\log P_{\boldsymbol{w}}(\boldsymbol{x},z,y)] \\ &= \mathbb{E}_{(\boldsymbol{x},y)}\mathbb{E}_{P_{z|\boldsymbol{x},y,\boldsymbol{w}_t}}[\log P(\boldsymbol{x})P_{\boldsymbol{w}}(z|\boldsymbol{x})P(y|\boldsymbol{x},z)] \\ &= \mathbb{E}_{(\boldsymbol{x},y)}\mathbb{E}_{P_{z|\boldsymbol{x},y,\boldsymbol{w}_t}}[\log P_{\boldsymbol{w}}(z|x)] + \text{const.} \\ &= \mathbb{E}[\sum_{i\in[k-1]} p_{\boldsymbol{w}_t}^{(i)}(\boldsymbol{w}_i^\top \boldsymbol{x}) - (1 + \sum_{i\in[k-1]} e^{\boldsymbol{w}_i^\top \boldsymbol{x}})] \\ &\qquad + \text{const.} \end{aligned} \tag{2.7}$$

where const refers to terms not depending on $\boldsymbol{w}$, $P_{\boldsymbol{w}}(z=i|x) = \exp(\boldsymbol{w}_i^\top \boldsymbol{x})/\sum_j \exp(\boldsymbol{w}_j^\top \boldsymbol{x})$ and $p_{\boldsymbol{w}_t}^{(i)} \triangleq \mathbb{P}(z=i|\boldsymbol{x},y,\boldsymbol{w}_t)$ corresponds to the posterior probability for the

i^{th} expert, given by

$$p^{(i)}_{\boldsymbol{w}_t} = \frac{p_{i,t}(\boldsymbol{x})\mathcal{N}(y|g(\boldsymbol{a}_i^\top \boldsymbol{x}), \sigma^2)}{\sum_{j\in[k]} p_{j,t}(\boldsymbol{x})\mathcal{N}(y|g(\boldsymbol{a}_j^\top \boldsymbol{x}), \sigma^2)}, \tag{2.8}$$
$$p_{i,t}(\boldsymbol{x}) = \frac{e^{(\boldsymbol{w}_t)_i^\top \boldsymbol{x}}}{1+\sum_{j\in[k-1]} e^{(\boldsymbol{w}_t)_j^\top \boldsymbol{x}}}.$$

In Equation 2.7, the expectation is with respect to the true distribution of $(\boldsymbol{x}, y)$, given by Equation 2.1. Thus the EM can be viewed as a deterministic procedure which maps $\boldsymbol{w}_t \mapsto M(\boldsymbol{w}_t)$ where

$$M(\boldsymbol{w}) = \text{argmax}_{\boldsymbol{w}'\in\Omega} Q(\boldsymbol{w}'|\boldsymbol{w}).$$

When the estimated regressors $\boldsymbol{a}_i$ equal the true parameters $\boldsymbol{a}_i^*$, it follows from the self-consistency property of the EM that the true parameter $\boldsymbol{w}^*$ is a fixed-point for the EM operator M, i.e. $M(\boldsymbol{w}^*) = \boldsymbol{w}^*$ [37]. However, this does not guarantee that EM converges to $\boldsymbol{w}^*$. In the following theorem, we show that even when the regressors are known approximately, EM algorithm converges to the true gating parameters at a geometric rate upto an additive error, under *global* initializations. For the error metric, we define $\|\boldsymbol{w} - \boldsymbol{w}'\| \triangleq \max_{i\in[k-1]} \|\boldsymbol{w}_i - \boldsymbol{w}'_i\|_2$ for any $\boldsymbol{w}, \boldsymbol{w}' \in \Omega$. We assume that $R = 1$ for simplicity. (Our results extend straightforwardly to general R).

Theorem 2. *Let $\varepsilon > 0$ be such that $\max_i \|\boldsymbol{a}_i - \boldsymbol{a}_i^*\|_2 = \sigma^2\varepsilon$. There exists a constant $\sigma_0 > 0$ such that whenever $0 < \sigma < \sigma_0$, for any random initialization $\boldsymbol{w}_0 \in \Omega$, the population-level EM updates on the gating parameter $\{\boldsymbol{w}\}_{t\geq 0}$ converge almost geometrically to the true parameter $\boldsymbol{w}^*$ upto an additive error, i.e.*

$$\|\boldsymbol{w}_t - \boldsymbol{w}^*\| \leq (\kappa_\sigma)^t \|\boldsymbol{w}_0 - \boldsymbol{w}^*\| + \kappa\varepsilon \sum_{i=0}^{t-1} \kappa_\sigma^i,$$

where κ_σ, κ are dimension-independent constant depending on g and σ such that $\kappa_\sigma \xrightarrow{\sigma\to 0} 0$ and $\kappa \leq (k-1)\frac{\sqrt{6(2+\sigma^2)}}{2}$ for $g =$linear, sigmoid and ReLU.

Remark. In the M-step of the EM algorithm, the next iterate is chosen so that the function $Q(\cdot|\boldsymbol{w}_t)$ is maximized. Instead we can perform an ascent step in the direction of the gradient of $Q(\cdot|\boldsymbol{w}_t)$ to produce the next iterate, i.e. $\boldsymbol{w}_{t+1} = \Pi_\Omega(\boldsymbol{w}_t + \alpha\nabla Q(\boldsymbol{w}_t|\boldsymbol{w}_t))$, where $\Pi_\Omega(\cdot)$ is the projection operator.

This variant of EM algorithm is known as *Gradient EM* which also enjoys similar convergence guarantees.

2.5 Experiments

In this section, we empirically validate our algorithm in various settings and compare its performance to that of EM on both synthetic and real world datasets [1]. In both the scenarios, we found that our algorithm consistently outperforms the existing approaches. For the tensor decomposition in our Algorithm 1, we use the Orth-ALS package by [38]. In all the synthetic experiments, we first draw the regressors $\{\boldsymbol{a}_i^*\}_{i=1}^k$ i.i.d uniformly from the unit sphere $\mathbb{S}^{d-1}$. The input distribution $P_{\boldsymbol{x}}$ and the generation of $\boldsymbol{w}_i^*$'s are detailed for each experiment. Then the labels y_i are generated according to the true k-MoE model in Equation 2.1 for linear activation.

2.5.1 Non-gaussian inputs

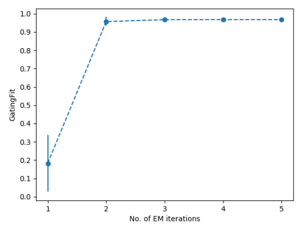
In this section we let the input distribution to be mixtures of Gaussians (GMM). We let $k = 2, d = 10$ and $\sigma = 0.1$. The gating parameter $\boldsymbol{w}^* \in \mathbb{R}^{10}$ is uniformly chosen from the unit sphere $\mathbb{S}^9$. To generate the input features, we first randomly draw $\mu_1, \mu_2 \in \mathbb{S}^9$, and generate n i.i.d. samples $\boldsymbol{x}_i \sim p\mathcal{N}(\mu_1, I_d) + (1-p)\mathcal{N}(\mu_2, I_d)$, where $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. Here $n = 2000$. Since $\boldsymbol{x}$ is a 2-GMM, its score functions $\mathcal{S}_3(\boldsymbol{x}), \mathcal{S}_2(\boldsymbol{x})$ are computed using the densities of Gaussian mixtures [39]. To gauge the performance of our algorithm, we measure the correlation of our learned parameters $\boldsymbol{a}_1, \boldsymbol{a}_2$ and $\boldsymbol{w}$ with the ground truth, i.e.

$$\mathsf{Regressor\ Fit}(\boldsymbol{a}_1, \boldsymbol{a}_2) = \max_{\pi} \min_{i \in \{1,2\}} |\langle \boldsymbol{a}_{\pi(i)}, \boldsymbol{a}_i^* \rangle|, \tag{2.9}$$

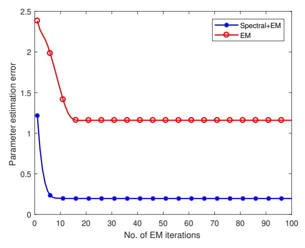
where $\pi : \{1,2\} \to \{1,2\}$ is a permutation. Similarly, for the gating parameter, we define

$$\mathsf{Gating\ Fit}(\boldsymbol{w}) = |\langle \boldsymbol{w}, \boldsymbol{w}^* \rangle|. \tag{2.10}$$

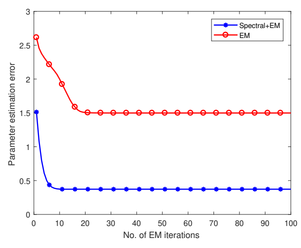
[1]Codes are available at this repository `https://github.com/Ashokvardhan/Breaking-the-gridlock-in-MoE-Consistent-and-Efficient-Algorithms`MoE codes.



(a) Non-orthogonality



(b) $k = 3$



(c) $k = 4$

Figure 2.3: (a): GatingFit for our algorithm under non-orthogonality setting. (b),(c): Estimation error $\mathcal{E}(\boldsymbol{A}, \boldsymbol{W})$ of our algorithm vs. joint-EM algorithm. Our algorithm is significantly better than the joint-EM under random initializations.

Here we assume that all the parameters are unit-normalized. The closer the values of fit are to one, the closer the learnt parameters are to the ground truth. As shown in Table 2.1, our algorithms are able to learn the ground truth very accurately in a variety of settings, as indicated by the measured fit. This highlights the fact that our algorithms are robust to the input distributions.

2.5.2 Non-orthogonal parameters

In this section we verify that our algorithms still work well in practice even under the relaxation of Assumption 3. For the experiments, we consider the similar setting as before with $k = 2, d = 10, \sigma = 0.1$ and the gating parameter $\boldsymbol{w}^*$ is drawn uniformly from $\mathbb{S}^9$ without the orthogonality restriction. We let $\boldsymbol{x}_i \overset{i.i.d.}{\sim} \mathcal{N}(0, I_d)$. We choose $n = 2000$. We use RegressorFit and GatingFit defined in Equation 2.9 and Equation 2.10 respectively, as our performance metrics. From Table 2.2, we can see that the performance of our algorithms is almost the same across both the settings. In both the scenarios, our fit is consistently greater than 0.9.

In Figure 2.3a, we plotted $\mathsf{GatingFit}(\boldsymbol{w}_t)$ vs. the number of iterations t, as $\boldsymbol{w}_t$ is updated according to Algorithm 2, over 10 independent trials. We observe that the learned parameters converge to the true parameters in less than five iterations.

Table 2.1: Fit of our learned parameters for non-Gaussian inputs

	$p = 0.1$	$p = 0.3$	$p = 0.5$	$p = 0.7$	$p = 0.9$
Regressor Fit	0.93 ± 0.06	0.94 ± 0.02	0.92 ± 0.04	0.92 ± 0.02	0.91 ± 0.06
Gating Fit	0.9 ± 0.1	0.97 ± 0.01	0.93 ± 0.04	0.96 ± 0.03	0.97 ± 0.01

Table 2.2: Performance of our algorithm under orthogonal and non-orthogonal settings

	Regressor Fit	Gating Fit
Non-orthogonal	0.9 ± 0.08	0.96 ± 0.02
Orthogonal	0.93 ± 0.03	0.96 ± 0.03

2.5.3 Comparison to joint-EM

Here we compare the performance of our algorithm with that of the joint-EM. We let the number of mixture components be $k = 3$ and $k = 4$. We let $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$ and the gating parameters are drawn uniformly from $\mathbb{S}^9$. If $\boldsymbol{A} = [\boldsymbol{a}_1 \ldots \boldsymbol{a}_k]$ and $\boldsymbol{W} = [\boldsymbol{w}_1 \ldots \boldsymbol{w}_{k-1} \; 0]$ denote the estimated expert and gating parameters respectively, our evaluation metric is $\mathcal{E}$, the Frobenious norm of the parameter error accounting for the best possible permutation $\pi : [k] \to [k]$, i.e. $\mathcal{E}(\boldsymbol{A}, \boldsymbol{W}) = \inf_\pi \|\boldsymbol{A} - \boldsymbol{A}^*_\pi\|_F + \|\boldsymbol{W} - \boldsymbol{W}^*_\pi\|_F$, where $\boldsymbol{A}^*_\pi = [\boldsymbol{a}^*_{\pi(1)} \ldots \boldsymbol{a}^*_{\pi(k)}]$ denotes the permuted regression parameter matrix and similarly for $\boldsymbol{W}^*_\pi$. In Figure 2.3b and Figure 2.3c, we compare the performance of our algorithm with the joint-EM algorithm for $n = 8000, d = 10, \sigma = 0.5$. The plotted estimation error $\mathcal{E}(\boldsymbol{A}, \boldsymbol{W})$ is averaged for 10 trials. It is clear that our algorithm is able to recover the true parameters thus resulting in much smaller parameter error than the joint-EM which often gets stuck in local optima. In addition, our algorithm is able to learn these parameters in very few iterations, often less than 10 iterations. We also find that our algorithm consistently outperforms the joint-EM for different choices of non-linearities, number of samples, number of mixtures, etc. Note that the above error metric $\mathcal{E}(\boldsymbol{A}, \boldsymbol{W})$ is close to zero if and only if Regressor Fit and Gating Fit is close to one.

Chapter 3

Mixture-of-Experts: Learning via Gradient Descent

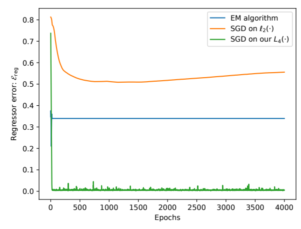
3.1 Connection between k-MoE and other popular models

In the previous section, we have seen how to learn the MoE parameters in a consistent way using a combination of spectral methods and the EM algorithm. However, a major drawback is that this approach requires specially crafted algorithms for learning each of these two sets of parameters. In addition, this lacks finite sample guarantees. Since SGD and its variants remain the de facto algorithms for training neural networks because of their practical advantages, and inspired by the successes of these gradient-descent based algorithms in finding global minima in a variety of non-convex problems, we ask the following question:

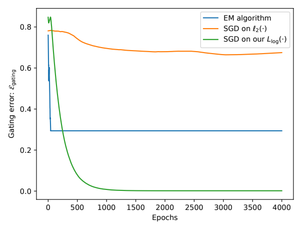
Question 2. *How do we design objective functions amenable to efficient optimization techniques, such as SGD, with provable learning guarantees for MoE?*

In this section, we address this question in a principled manner and propose two non-trivial non-convex loss functions $L_4(\cdot)$ and $L_{\log}(\cdot)$ to learn the regressors and the gating parameters respectively. In particular, our loss functions possess nice landscape properties such as local minima being global and the global minima corresponding to the ground truth parameters. We also show that gradient descent on our losses can recover the true parameters with *global/random initializations*. To the best of our knowledge, ours is the first GD based approach with finite sample guarantees to learn the parameters of MoE. We summarize our main contributions below:

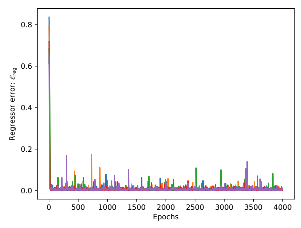
- **Optimization landscape design with desirable properties:** We design *two non-trivial* loss functions $L_4(\cdot)$ and $L_{\log}(\cdot)$ to learn the



(a) Regressor error



(b) Gating error



(c) $L_4(\cdot)$ over different initializations

Figure 3.1: Our proposed losses L_4 (defined in Equation 3.2) and $L_{\log}$ (defined in Equation 3.4) to learn the respective regressor and gating parameters of a MoE model in Equation 2.1 achieve much better empirical results than the standard methods.

regressors and the gating parameters of a MoE separately. We show that our loss functions have nice landscape properties and are amenable to simple local-search algorithms. In particular, we show that SGD on our novel loss functions recovers the parameters with *global/random* initializations.

- **First sample complexity results:** We also provide the first sample complexity results for MoE. We show that our algorithms can recover the true parameters with accuracy ε and with high probability, when provided with samples *polynomial* in the dimension d and $1/\varepsilon$.

Overview. The rest of this section is organized as follows: In Section 3.2 we design two novel loss functions to learn the respective regressors and gating parameters of a MoE and present our theoretical guarantees. In Section 3.3, we empirically validate that our proposed losses perform much better than the current approaches on a variety of settings.

3.2 Optimization landscape design for MoE

In this section, we focus on the learnability of the MoE model and design two novel loss functions for learning the regressors and the gating parameters separately.

3.2.1 Loss function for regressors: L_4

To motivate the need for loss function design in a MoE, first we take a moment to highlight the issues with the traditional approach of using the mean square loss ℓ_2. If (x, y) are generated according to the ground-truth MoE model in Equation 2.1, $\ell_2(\cdot)$ computes the quadratic cost between the expected predictions $\hat{y}$ and the ground-truth y, i.e.

$$\ell_2(\{a_i\}, \{w_i\}) = \mathbb{E}_{(x,y)} \|\hat{y}(x) - y\|^2,$$

where $\hat{y}(x) = \sum_i \text{softmax}_i(w_1^\top x, \ldots, w_{k-1}^\top x, 0)\ g(\boldsymbol{a}_i^\top \boldsymbol{x})$ is the predicted output, and $\{a_i\}, \{w_i\}$ denote the respective regressors and gating parameters. It is well-known that this mean square loss is prone to bad local minima as demonstrated empirically in the earliest work of [19] (we verify this in Section 3.3 too), which also emphasized the importance of the right objective function to learn the parameters. Note that the bad landscape of ℓ_2 is not just unique to MoE but also widely observed in the context of training neural network parameters [40]. In the one-hidden-layer NN setting, some recent works [41, 42] addressed this issue by designing new loss functions with good landscape properties so that standard algorithms like SGD can provably learn the parameters. However these methods do not generalize to the MoE setting since they crucially rely on the fact that the coefficients z_i appearing in front of the activation terms $g(\langle a_i^*, x\rangle)$ in Equation 2.1, which correspond to the linear layer weights in NN, are constant. Such an assumption does not hold in the context of MoEs because the gating probabilities depend on x in a parametric way through the softmax function and hence introducing the coupling between w_i^* and a_i^* (a similar observation was noted in [43] in the context of spectral methods).

In order to address the aforementioned issues, inspired by the works of [41] and [42], we design a novel loss function $L_4(\cdot)$ to learn the regressors first. Our loss function depends on two distinct special transformations on both the input $x \in \mathbb{R}^d$ and the output $y \in \mathbb{R}$. For the output, we consider the following transformations:

$$\mathcal{Q}_4(y) \triangleq y^4 + \alpha y^3 + \beta y^2 + \gamma y, \ \ \mathcal{Q}_2(y) \triangleq y^2 + \delta y, \tag{3.1}$$

where the set of coefficients $(\alpha, \beta, \gamma, \delta)$ are dependent on the choice of non-linearity g and noise variance σ^2. These are obtained by solving a simple linear system. For the special case $g = Id$, which corresponds to linear activations, the Quartic transform is $\mathcal{Q}_4(y) = y^4 - 6y^2(1+\sigma^2) + 3 + 3\sigma^4 - 6\sigma^2$ and the Quadratic transform is $\mathcal{Q}_2(y) = y^2 - (1+\sigma^2)$. For the input $\boldsymbol{x}$, we assume that $x \sim \mathcal{N}(0, I_d)$, and for any two fixed $u, v \in \mathbb{R}^d$, we consider the projections of multivariate-Hermite polynomials [39, 44, 45] along these two vectors, i.e.

$$\begin{aligned}
t_3(\boldsymbol{u}, \boldsymbol{x}) &= \frac{(\boldsymbol{u}^\top \boldsymbol{x})^2 - \|\boldsymbol{u}\|^2}{c'_{g,\sigma}}, \\
t_2(\boldsymbol{u}, \boldsymbol{x}) &= \frac{(\boldsymbol{u}^\top \boldsymbol{x})^4 - 6\|\boldsymbol{u}\|^2(\boldsymbol{u}^\top \boldsymbol{x})^2 + 3\|\boldsymbol{u}\|^4}{c_{g,\sigma}}, \\
t_1(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}) &= ((\boldsymbol{u}^\top \boldsymbol{x})^2(\boldsymbol{v}^\top \boldsymbol{x})^2 - \|\boldsymbol{u}\|^2(\boldsymbol{v}^\top \boldsymbol{x})^2 \\
&\quad - 4(\boldsymbol{u}^\top \boldsymbol{x})(\boldsymbol{v}^\top \boldsymbol{x})(\boldsymbol{u}^\top \boldsymbol{v}) - \|\boldsymbol{v}\|^2(\boldsymbol{u}^\top \boldsymbol{x})^2 \\
&\quad + \|\boldsymbol{u}\|^2\|\boldsymbol{v}\|^2 + 2(\boldsymbol{u}^\top \boldsymbol{v})^2)/c_{g,\sigma},
\end{aligned}$$

where $c_{g,\sigma}$ and $c'_{g,\sigma}$ are two non-zero constants depending on g and σ. These transformations (t_1, t_2, t_3) on the input x and $(\mathcal{Q}_4, \mathcal{Q}_2)$ on the output y can be viewed as extractors of higher order information from the data. The utility of these transformations is concretized in Theorem 3 through the loss function defined below. Denoting the set of our regression parameters by the matrix $\boldsymbol{A}^\top = [\boldsymbol{a}_1|\boldsymbol{a}_2|\ldots|\boldsymbol{a}_k] \in \mathbb{R}^{d\times k}$, we now define our objective function $L_4(\boldsymbol{A})$ as

$$\begin{aligned}
&L_4(\boldsymbol{A}) \\
&\triangleq \sum_{\substack{i,j\in[k] \\ i\neq j}} \mathbb{E}[\mathcal{Q}_4(y) t_1(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{x})] - \mu \sum_{i\in[k]} \mathbb{E}[\mathcal{Q}_4(y) t_2(\boldsymbol{a}_i, \boldsymbol{x})] \\
&+ \lambda \sum_{i\in[k]} (\mathbb{E}[\mathcal{Q}_2(y) t_3(\boldsymbol{a}_i, \boldsymbol{x})] - 1)^2 + \frac{\delta}{2}\|\boldsymbol{A}\|_F^2, \qquad (3.2)
\end{aligned}$$

where $\mu, \lambda, \delta > 0$ are some positive regularization constants. Notice that L_4 is defined as an expectation of terms involving the data transformations: $\mathcal{Q}_4, \mathcal{Q}_2, t_1, t_2$, and t_3. Hence its gradients can be readily computed from finite samples and is amenable to standard optimization methods such as SGD for learning the parameters. Moreover, the following theorem highlights that the landscape of L_4 does not have any spurious local minima.

Theorem 3 (Landscape analysis for learning regressors)**.** *Under the mild* *technical assumptions of [43], the loss function L_4 does not have any spurious local minima. More concretely, let $\varepsilon > 0$ be a given error tolerance. Then we can choose the regularization constants μ, λ and the parameters ε, τ such that if A satisfies*

$$\|\nabla L_4(\boldsymbol{A})\|_2 \leq \varepsilon, \quad \nabla^2 L_4(\boldsymbol{A}) \succcurlyeq -\tau/2,$$

then $(\boldsymbol{A}^\dagger)^\top = \boldsymbol{P}\boldsymbol{D}\Gamma\boldsymbol{A}^* + \boldsymbol{E}$, where $\boldsymbol{D}$ is a diagonal matrix with entries close to 1, Γ is a diagonal matrix with $\Gamma_{ii} = \sqrt{\mathbb{E}[p_i^*(\boldsymbol{x})]}$, $\boldsymbol{P}$ is a permutation matrix and $\|\boldsymbol{E}\| \leq \varepsilon_0$. Hence every approximate local minimum is ε-close to the global minimum.

Intuitions behind the theorem and the special transforms: While the transformations and the loss L_4 defined above may appear non-intuitive at first, the key observation is that L_4 can be viewed as a fourth-order polynomial loss in the parameter space, i.e.

$$\begin{aligned} &L_4(\boldsymbol{A}) \\ &= \sum_{m\in[k]} \mathbb{E}[p_m^*(\boldsymbol{x})] \sum_{\substack{i\neq j\\ i,j\in[k]}} \langle \boldsymbol{a}_m^*, \boldsymbol{a}_i\rangle^2 \langle \boldsymbol{a}_m^*, \boldsymbol{a}_j\rangle^2 \\ &- \mu \sum_{m,i\in[k]} \mathbb{E}[p_m^*(\boldsymbol{x})]\langle \boldsymbol{a}_m^*, \boldsymbol{a}_i\rangle^4 \\ &+ \lambda \sum_{i\in[k]} \Big(\sum_{m\in[k]} \mathbb{E}[p_m^*(\boldsymbol{x})]\langle \boldsymbol{a}_m^*, \boldsymbol{a}_i\rangle^2 - 1\Big)^2 + \frac{\delta}{2}\|\boldsymbol{A}\|_F^2, \end{aligned} \tag{3.3}$$

where p_i^* refers to the softmax probability for the i^{th} label with true gating parameters, i.e. $p_i^*(x) = \text{softmax}_i(\langle w_1^*, x\rangle, \ldots, \langle w_{k-1}^*, x\rangle, 0)$. This alternate characterization of $L_4(\cdot)$ in Equation 3.3 is the crucial step towards proving Theorem 3. Hence these specially designed transformations on the data (x, y) help us to achieve this objective. Given this viewpoint, we utilize tools from [41] where a similar loss involving fourth-order polynomials were analyzed in the context of 1-layer ReLU network to prove the desired landscape properties for L_4. The full details behind the proof are provided in Appendix B.2. Moreover, in Section 3.3 we empirically verify that the technical assumptions are only needed for the theoretical results and that our algorithms are robust to these assumptions and work equally well even when we relax them.

In the finite sample regime, we replace the population expectations in

Equation 3.2 with sample average to obtain the empirical loss $\hat{L}$. The following theorem establishes that $\hat{L}$ too inherits the same landscape properties of L when provided enough samples.

Theorem 4 (Finite sample landscape)**.** *There exists a polynomial* $\mathrm{poly}(d, 1/\varepsilon)$ *such that whenever* $n \geq \mathrm{poly}(d, 1/\varepsilon)$*,* $\hat{L}$ *inherits the same landscape properties as that of* L *established in Theorem 3 with high probability. Hence stochastic gradient descent on* $\hat{L}$ *converges to an approximate local minima which is also close to a global minimum in time polynomial in* $d, 1/\varepsilon$*.*

Remark 1. Notice that the parameters $\{\boldsymbol{a}_i\}$ learnt through SGD are some permutation of the true parameters $\boldsymbol{a}_i^*$ upto sign flips. This sign ambiguity can be resolved using existing standard procedures such as Algorithm 1 in [41]. In the remainder of the paper, we assume that we know the regressors upto some error $\varepsilon_{\text{reg}} > 0$ in the following sense: $\max_{i \in [k]} \|a_i - a_i^*\| = \sigma^2 \varepsilon_{\text{reg}}$.

3.2.2 Loss function for gating parameters: $L_{\log}$

In the previous section, we have established that we can learn the regressors $\boldsymbol{a}_i^*$ upto small error using SGD on the loss function L_4. Now we are interested in answering the following question: Can we design a loss function amenable to efficient optimization algorithms such as SGD with recoverable guarantees to learn the gating parameters?

In order to gain some intuition towards addressing this question, consider the simplified setting of $\sigma = 0$ and $\boldsymbol{A} = \boldsymbol{A}^*$. In this setting, we can see from Equation 2.1 that the output y equals one of the activation values $g(\langle \boldsymbol{a}_i^*, \boldsymbol{x} \rangle)$, for $i \in [k]$, with probability 1. Since we already have access to the true parameters, i.e. $A = \boldsymbol{A}^*$, we can see that we can exactly recover the hidden latent variable z, which corresponds to the chosen hidden expert for each sample $(\boldsymbol{x}, y)$. Thus the problem of learning the *classifiers* $\boldsymbol{w}_i^*, \ldots, \boldsymbol{w}_{k-1}^*$ reduces to a multi-class classification problem with label z for each input $\boldsymbol{x}$ and hence can be efficiently solved by traditional methods such as logistic regression. It turns out that these observations can be formalized to deal with more general settings (where we only know the regressors approximately and the noise variance is not zero) and that the gradient descent on the log-likelihood loss achieves the same objective. Hence we use the negative log-likelihood function to learn the classifiers, i.e.

$$\begin{aligned}
&L_{\log}(\boldsymbol{W}, \boldsymbol{A}) \\
&\triangleq -\mathbb{E}_{(x,y)}[\log P_{y|\boldsymbol{x}}] \qquad (3.4)\\
&= -\mathbb{E} \log \left(\sum_{i \in [k]} \frac{e^{\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle}}{\sum_{j \in [k]} e^{\langle \boldsymbol{w}_j, \boldsymbol{x} \rangle}} \cdot \mathcal{N}(y | g(\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle), \sigma^2) \right),
\end{aligned}$$

where $\boldsymbol{W}^\top = \left[\boldsymbol{w}_1 | \boldsymbol{w}_2 | \dots | \boldsymbol{w}_{k-1} \right]$. Note that the objective Equation 3.4 in not convex in the gating parameters W whenever $\sigma \neq 0$. We omit the input distribution P_x from the above negative log-likelihood since it does not depend on any of the parameters. We now define the domain of the gating parameters Ω as

$$\boldsymbol{W} \in \Omega \triangleq \{ \boldsymbol{W} \in \mathbb{R}^{(k-1) \times d} : \|w_i\|_2 \leq R, \forall i \in [k-1] \},$$

for some fixed $R > 0$. Without loss of generality, we assume that $\boldsymbol{w}_k = 0$. Since we know the regressors approximately from the previous stage, i.e. $\boldsymbol{A} \approx \boldsymbol{A}^*$, we run gradient descent only for the classifier parameters keeping the regressors fixed, i.e.

$$\boldsymbol{W}_{t+1} = \Pi_\Omega(\boldsymbol{W}_t - \alpha \nabla_{\boldsymbol{W}} L_{\log}(\boldsymbol{W}_t, \boldsymbol{A})),$$

where $\alpha > 0$ is a suitably chosen learning-rate, $\Pi_\Omega(\boldsymbol{W})$ denotes the projection operator which maps each row of its input matrix onto the ball of radius R, and $t > 0$ denotes the iteration step. In a more succinct way, we write

$$\begin{aligned}
\boldsymbol{W}_{t+1} &= G(\boldsymbol{W}_t, \boldsymbol{A}), \\
G(\boldsymbol{W}, \boldsymbol{A}) &\triangleq \Pi_\Omega(\boldsymbol{W} - \alpha \nabla_{\boldsymbol{W}} L_{\log}(\boldsymbol{W}, \boldsymbol{A})).
\end{aligned}$$

Note that $G(W, A)$ denotes the projected gradient descent operator on W for fixed A. In the finite sample regime, we define our loss $L_{\log}^{(n)}(\boldsymbol{W}, \boldsymbol{A})$ as the finite sample counterpart of Equation 3.4 by taking empirical expectations. Accordingly, we define the gradient operator $G_n(\boldsymbol{W}, \boldsymbol{A})$ as

$$G_n(\boldsymbol{W}, \boldsymbol{A}) \triangleq \Pi_\Omega(\boldsymbol{W} - \alpha \nabla_{\boldsymbol{W}} L_{\log}^{(n)}(\boldsymbol{W}, \boldsymbol{A})).$$

In this paper, we analyze a sample-splitting version of the gradient descent, where given the number of samples n and the iterations T, we first split the data into T subsets of size $\lfloor n/T \rfloor$, and perform iterations on fresh batch of samples, i.e. $\boldsymbol{W}_{t+1} = G_{n/T}(\boldsymbol{W}_t, \boldsymbol{A})$. We use the norm $\|W - W^*\| = \max_{i \in [k-1]} \|w_i - w_i^*\|_2$ for our theoretical results. The following theorem establishes the almost geometric convergence of the population-gradient iterates under some high SNR conditions. The following results are stated for $R = 1$ for simplicity and also hold for any general $R > 0$.

Theorem 5 (GD convergence for classifiers)**.** *Assume that* $\max_{i \in [k]} \|a_i - a_i^*\|_2 = \sigma^2 \varepsilon_{\text{reg}}$. *Then there exists two positive constants* α_0 *and* σ_0 *such that for any step size* $0 < \alpha \leq \alpha_0$ *and noise variance* $\sigma^2 < \sigma_0^2$*, the population gradient descent iterates* $\{\boldsymbol{W}\}_{t \geq 0}$ *converge almost geometrically to the true parameter* $\boldsymbol{W}^*$ *for any randomly initialized* $W_0 \in \Omega$*, i.e.*

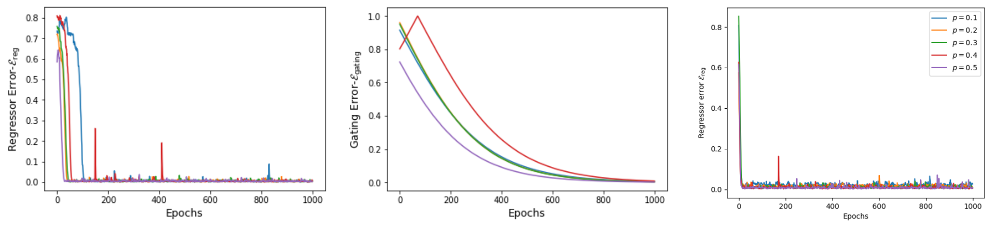
$$\|\boldsymbol{W}_t - \boldsymbol{W}^*\| \leq (\rho_\sigma)^t \|\boldsymbol{W}_0 - \boldsymbol{W}^*\| + \kappa \varepsilon_{\text{reg}} \sum_{\tau=0}^{t-1} (\rho_\sigma)^\tau,$$

where $(\rho_\sigma, \kappa) \in (0,1) \times (0, \infty)$ *are dimension-independent constants depending on* g, k *and* σ *such that* $\rho_\sigma = o_\sigma(1)$ *and* $\kappa = O_{k,\sigma}(1)$.

Proof. (Sketch) For simplicity, let $\varepsilon_{\text{reg}} = 0$. Then we can show that $G(\boldsymbol{W}^*, \boldsymbol{A}^*) = \boldsymbol{W}^*$ since $\nabla_{\boldsymbol{W}} L_{\log}(\boldsymbol{W} = \boldsymbol{W}^*, \boldsymbol{A}^*) = 0$. Then we capitalize on the fact that $G(\cdot, \boldsymbol{A}^*)$ is strongly convex with minimizer at $\boldsymbol{W} = \boldsymbol{W}^*$ to show the geometric convergence rate. The more general case of $\varepsilon_{\text{reg}} > 0$ is handled through perturbation analysis. $\square$

We conclude our theoretical discussion on MoE by providing the following finite sample complexity guarantees for learning the classifiers using the gradient descent in the following theorem, which can be viewed as a finite sample version of Theorem 5.

Theorem 6 (Finite sample complexity and convergence rates for GD)**.** *In addition to the assumptions of Theorem 5, assume that the sample size* n *is lower bounded as* $n \geq c_1 T d \log(\frac{T}{\delta})$. *Then the sample-gradient iterates*



(a) (b) (c)

Figure 3.2: (a), (b): Robustness to parameter orthogonality: Plots show performance over 5 different trials for our losses L_4 and L_{log} respectively. (c) Robustness to Gaussianity of input: Performance over various mixing probabilities p.

$\{\boldsymbol{W}^t\}_{t=1}^T$ *based on* n/T *samples per iteration satisfy the bound*

$$\|\boldsymbol{W}^t - \boldsymbol{W}^*\| \leq (\rho_\sigma)^t \|\boldsymbol{W}_0 - \boldsymbol{W}^*\| + \frac{1}{1-\rho_\sigma}\left(\kappa\varepsilon_{\text{reg}} + c_2\sqrt{\frac{dT\log(Tk/\delta)}{n}}\right)$$

with probability at least $1 - \delta$.

3.3 Experiments

In this section, we empirically validate the fact that running SGD on our novel loss functions L_4 and L_{log} achieves superior performance compared to the existing approaches. Moreover, we empirically show that our algorithms are robust to the technical assumptions made in Theorem 3 and that they achieve equally good results even when the assumptions are relaxed.

Data generation. For our experiments, we choose $d = 10$, $k \in \{2, 3\}$, $\boldsymbol{a}_i^* = \boldsymbol{e}_i$ for $i \in [k]$ and $\boldsymbol{w}_i^* = \boldsymbol{e}_{k+i}$ for $i \in [k-1]$, and $g = Id$. We generate the data $\{(\boldsymbol{x}_i, y_i)_{i=1}^n\}$ according to Equation 2.1 and using these ground-truth parameters. We chose $\sigma = 0.05$ for all of our experiments.

Error metric. If $\boldsymbol{A} \in \mathbb{R}^{k\times d}$ denotes the matrix of regressors where each row is of norm 1, we use the error metric $\mathcal{E}_{\text{reg}}$ to gauge the closeness of $\boldsymbol{A}$ to the ground-truth $\boldsymbol{A}^*$:

$$\mathcal{E}_{\text{reg}} \triangleq 1 - \max_{\pi \in S_k} \min_{i \in [k]} |\langle \boldsymbol{a}_i, \boldsymbol{a}^*_{\pi(i)} \rangle|,$$

where S_k denotes the set of all permutations on $[k]$. Note that $\mathcal{E}_{\text{reg}} \leq \varepsilon$ if and only if the learnt regressors have a minimum correlation of $1 - \varepsilon$ with the ground-truth parameters, upto a permutation. The error metric $\mathcal{E}_{\text{gating}}$ is defined similarly.

Results. In Figure 3.1, we choose $k = 3$ and compare the performance of our algorithm against existing approaches. In particular, we consider three methods: 1) EM algorithm, 2) SGD on the the classical ℓ_2-loss from Equation 3.1, and 3) SGD on our losses L_4 and $L_{\log}$. For all the methods, we ran 5 independent trials and plotted the mean error. Figure 3.1a highlights the fact that minimizing our loss function L_4 by SGD recovers the ground-truth regressors, whereas SGD on ℓ_2-loss as well as EM get stuck in local optima. For learning the gating parameters W using our approach, we first fix the regressors A at the values learnt using L_4, i.e. $A = \hat{A}$, where $\hat{A}$ is the converged solution for L_4. For ℓ_2 and the EM algorithm, the gating parameters W are learnt jointly with regressors A. Figure 3.1b illustrates the phenomenon that our loss $L_{\log}$ for learning the gating parameters performs considerably better than the standard approaches, as indicated in significant gaps between the respective error values. Finally, in Figure 3.1c we plot the regressor error for L_4 over 5 random initializations. We can see that we recover the ground truth parameters in all the trials, thus empirically corroborating our technical results in Section 3.2.

3.3.1 Robustness to technical assumptions

In this section, we verify numerically the fact that our algorithms work equally well in the absence of technical assumptions made in Section 3.2.

Relaxing orthogonality in Theorem 3. A key assumption in proving Theorem 3, adapted from [43], is that the set of regressors $\{a_i^*\}$ and set of gating parameters $\{w_i^*\}$ are orthogonal to each other. While this assumption is needed for the technical proofs, we now empirically verify that our conclusions still hold when we relax this. For this experiment, we choose $k = 2$ and let $(a_1^*, a_2^*) = (e_1, e_2)$. For the gating parameter $w^* \triangleq w_1^*$, we randomly generate it from uniform distribution on the d-dimensional unit sphere. In Figure A.2a and Figure A.2b, we plotted the individual parameter estimation error for 5 different runs for both of our losses L_4 and $L_{\log}$ for learning the regressors and the gating parameter respectively. We can see that our algorithms are still able to learn the true parameters even when the orthogonality assumption is relaxed.

Relaxing Gaussianity of the input. To demonstrate the robustness of our approach to the assumption that the input x is standard Gaussian, i.e. $x \sim \mathcal{N}(0, I_d)$, we generated x according to a mixture of two symmetric Gaussians each with identity covariances, i.e. $x \sim p\mathcal{N}(\mu, I_d) + (1-p)\mathcal{N}(-\mu, I_d)$, where $p \in [0, 1]$ is the mixing probability and $\mu \in \mathbb{R}^d$ is a fixed but randomly chosen vector. For various mixing proportions $p \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, we ran SGD on our loss L_4 to learn the regressors. Figure 3.2c highlights that we learn these ground truth parameters in all the settings.

Finally we note that in all our experiments, the loss L_4 seems to require a larger batch size (1024) for its gradient estimation while running SGD. However, with smaller batch sizes such as 128 we are still able to achieve similar performance but with more variance.

Chapter 4

KO Codes: Novel Neural Codes

4.1 Introduction

Physical layer communication underpins the information age (WiFi, cellular, cable and satellite modems). Codes, composed of encoder and decoder pairs, are the basic mathematical objects enabling reliable communication: encoder maps original data bits into a longer sequence, and decoders map the received sequence to the original bits. Reliability is precisely measured: bit error rate (BER) measures the fraction of input bits that were incorrectly decoded; block error rate (BLER) measures the fraction of times at least one of the original data bits was incorrectly decoded.

Landmark codes include Reed-Muller (RM), BCH, Turbo, LDPC and Polar codes [3]: each is a linear code and represents a mathematical breakthrough discovered over a span of six decades. The impact on humanity is huge: each of these codes has been used in global communication standards over the past six decades. These codes essentially operate at the information-theoretic limits of reliability over the additive white Gaussian noise (AWGN) channel, when the number of information bits is large, the so-called “large block length" regime. In the small and medium block length regimes, the state-of-the-art codes are *algebraic*: encoders and decoders are invented based on specific linear algebraic constructions over the binary and higher order fields and rings. Especially prominent binary algebraic codes are RM codes and closely related polar codes, whose encoders are recursively defined as Kronecker products of a simple linear operator and constitute the state of the art in small-to-medium block length regimes.

Inventing new codes is a major intellectual activity both in academia and the wireless industry; this is driven by emerging practical applications, e.g., low block length regime in Internet of Things [46]. The core challenge is that the space of codes is very vast and the sizes astronomical; for instance a rate 1/2 code over even 100 information bits involves designing 2^{100} codewords in a 200 dimensional space. Computationally efficient encoding and decoding procedures are a must, apart from high reliability. Thus, although a random code is information theoretically optimal, neither encoding nor decoding is computationally efficient. The mathematical landscape of computationally efficient codes has been plumbed over the decades by some of the finest mathematical minds, resulting in two distinct families of codes: *algebraic codes* (RM, BCH – focused on properties of polynomials) and *graph codes* (Turbo, LDPC – based on sparse graphs and statistical physics). The former is deterministic and involves discrete mathematics, while the latter harnesses randomness, graphs, and statistical physics to behave like a pseudorandom code. A major open question is the invention of new codes, and especially fascinating would be a family of codes outside of these two classes.

Our major result is the invention of a new family of codes, called KO codes, that have features of both code families: they are nonlinear generalizations of the Kronecker operation underlying the algebraic codes (e.g., Reed-Muller) parameterized by neural networks; the parameters are learnt in an end-to-end training paradigm in a data driven manner. Deep learning (DL) has transformed several domains of human endeavor that have traditionally relied heavily on mathematical ingenuity, e.g., game playing (AlphaZero [47]), biology (AlphaFold [48]), and physics (new laws [49]). Our results can be viewed as an added domain to the successes of DL in inventing mathematical structures.

A linear encoder is defined by a *generator matrix*, which maps information bits to a codeword. The RM and the Polar families construct their generator matrices by recursively applying the Kronecker product operation to a simple two-by-two matrix and then selecting rows from the resulting matrix. The careful choice in selecting these rows is driven by the desired algebraic structure of the code, which is central to achieving the large *minimum* pairwise distance between two codewords, a hallmark of the algebraic family. This encoder can be alternatively represented by a computation graph. The recursive Kronecker product corresponds to a complete binary tree, and row-selection corresponds to freezing a set of leaves in the tree, which we refer to as a “Plotkin tree", inspired by the pioneering construction in [50].

The Plotkin tree skeleton allows us to tailor a new neural network architecture: we expand the algebraic family of codes by replacing the (linear) Plotkin construction with a non-linear operation parametrized by neural networks. The parameters are discovered by training the encoder with a matching decoder, that has the matching Plotkin tree as a skeleton, to minimize the error rate over (the unlimited) samples generated on AWGN channels.

Algebraic and the original RM codes promise a large worst-case pairwise distance [51]. This ensures that RM codes achieve capacity in the large block length limit [52]. However, for short block lengths, they are too conservative as we are interested in the average-case reliability. This is the gap KO codes exploit: we seek a better average-case reliability and not the minimum pairwise distance.

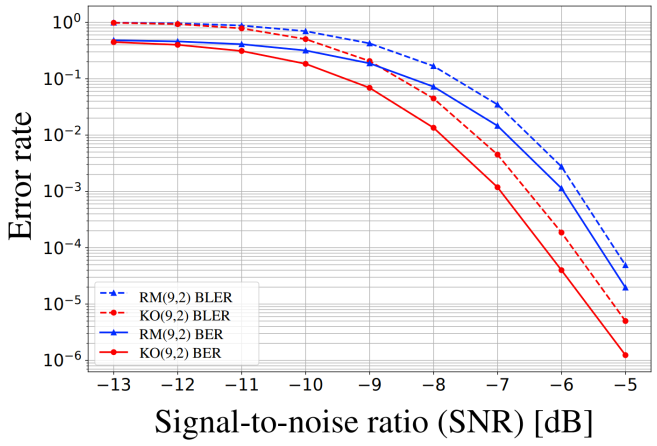


Figure 4.1: KO$(9,2)$, discovered by training a neural network with a carefully chosen architecture in Section 4.3, significantly improves upon state-of-the-art RM$(9,2)$ both in BER and BLER. (For both codes, the code block length is $2^9 = 512$ and the number of transmitted message bits is $\binom{9}{0} + \binom{9}{1} + \binom{9}{2} = 55$. Also, both codes are decoded using successive cancellation decoding with similar decoding complexity)

Figure 4.1 illustrates the gain for the example of RM$(9,2)$ code. Using the Plotkin tree of RM$(9,2)$ code as a skeleton, we design the KO$(9,2)$ code architecture and train on samples simulated over an AWGN channel. We discover a novel non-linear code and a corresponding efficient decoder that improves significantly over the RM$(9,2)$ code baseline, assuming both codes are decoded using successive cancellation decoding with similar decoding complexity. Analyzing the pairwise distances between two codewords reveals a surprising fact. The histogram for KO code nearly matches that of a random

Gaussian codebook. The skeleton of the architecture from an algebraic family of codes, the training process with a variation of the stochastic gradient descent, and the simulated AWGN channel have worked together to discover a novel family of codes that harness the benefits of both algebraic and pseudorandom constructions.

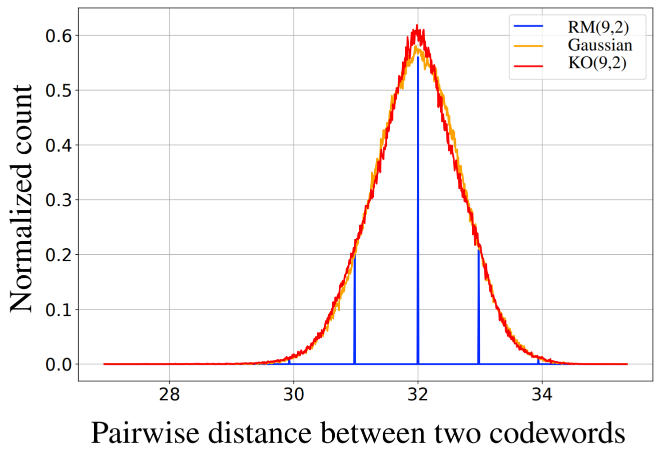


Figure 4.2: Histogram of pairwise distances between codewords of the KO$(9, 2)$ code shows a strong resemblance to that of the Gaussian codebook, unlike the classical Reed-Muller code RM$(9, 2)$.

In summary, we make the following contributions: We introduce novel neural network architectures for the (encoder, decoder) pair that generalizes the Kronecker operation central to RM/Polar codes. We propose training methods that discover novel non-linear codes when trained over AWGN and provide empirical results showing that this family of non-linear codes improves significantly upon the baseline code it was built on (both RM and Polar codes) whilst having the same encoding and decoding complexity. Interpreting the pairwise distances of the discovered codewords reveals that a KO code mimics the distribution of codewords from the random Gaussian codebook, which is known to be reliable but computationally challenging to decode. The decoding complexities of KO codes are $O(n \log n)$ where n is the block length, matching that of efficient decoders for RM and Polar codes.

We highlight that the design principle of KO codes serves as a general recipe to discover new family of non-linear codes improving upon their linear counterparts. In particular, the construction is not restricted to a specific decoding algorithm, such as successive cancellation (SC). In this paper, we focus on the SC decoding algorithm since it is one of the most efficient decoders for the RM and Polar family. At this decoding complexity, i.e. $O(n \log n)$, our results demonstrate that we achieve significant gain over these codes. Our preliminary results show that KO codes achieve similar gains over the RM codes, when both are decoded with list-decoding. We refer to §C.2 for more details. Designing KO-inspired codes to improve upon the RPA decoder for RM codes (with complexity $O(n^r \log n)$ [53]), and the list-decoded Polar codes (with complexity $O(Ln \log n)$ [54]) where L is the list size, are promising active research directions, and outside the scope of this paper.

4.2 Problem formulation and background

We formally define the channel coding problem and provide background on Reed-Muller codes, the inspiration for our approach. Our notation is the following. We denote Euclidean vectors by bold face letters like $\boldsymbol{m}, \boldsymbol{L}$, etc. For $\boldsymbol{L} \in \mathbb{R}^n$, $\boldsymbol{L}_{k:m} \triangleq (L_k, \ldots, L_m)$. If $\boldsymbol{v} \in \{0,1\}^n$, we define the operator $\oplus_{\boldsymbol{v}}$ as $\boldsymbol{x} \oplus_{\boldsymbol{v}} \boldsymbol{y} \triangleq \boldsymbol{x} + (-1)^{\boldsymbol{v}} \boldsymbol{y}$.

4.2.1 Channel coding

Let $\boldsymbol{m} = (m_1, \ldots, m_k) \in \{0,1\}^k$ denote a block of *information/message bits* that we want to transmit. An encoder $g_\theta(\cdot)$ is a function parametrized by θ that maps these information bits into a binary vector $\boldsymbol{x}$ of length n, i.e. $\boldsymbol{x} = g_\theta(\boldsymbol{m}) \in \{0,1\}^n$. The *rate* $\rho = k/n$ of such a code measures how many bits of information we are sending per channel use. These codewords are transformed into real (or complex) valued signals, called modulation, before being transmitted over a channel. For example, Binary Phase Shift Keying (BPSK) modulation maps each $x_i \in \{0,1\}$ to $1 - 2x_i \in \{\pm 1\}$ up to a universal scaling constant for all $i \in [n]$. Here, we do not strictly separate encoding from modulation and refer to both binary encoded symbols and real-valued transmitted symbols as *codewords*. The codewords also satisfy either a hard or soft power constraint. Here we consider the hard power constraint, i.e., $\|x\|^2 = n$.

Upon transmission of this codeword $\boldsymbol{x}$ across a noisy channel $P_{Y|X}(\cdot|\cdot)$, we receive its corrupted version $\boldsymbol{y} \in \mathbb{R}^n$. The decoder $f_\phi(\cdot)$ is a function parametrized by ϕ that subsequently processes the received vector $\boldsymbol{y}$ to estimate the information bits $\hat{\boldsymbol{m}} = f_\phi(\boldsymbol{y})$. The closer $\hat{\boldsymbol{m}}$ is to $\boldsymbol{m}$, the more reliable the transmission. An error metric, such as Bit-Error-Rate (BER) or Block-Error-Rate (BLER), gauges the performance of the encoder-decoder pair (g_θ, f_ϕ). Note that BER is defined as $\text{BER} \triangleq (1/k)\sum_i \mathbb{P}(\hat{m}_i \neq m_i)$, whereas $\text{BLER} \triangleq \mathbb{P}(\hat{\boldsymbol{m}} \neq \boldsymbol{m})$.

The design of good codes given a channel and a fixed set of code parameters (k, n) can be formulated as:

$$(\theta, \phi) \quad \in \quad \arg\min_{\theta,\phi}\ \text{BER}(g_\theta, f_\phi)\ , \tag{4.1}$$

which is a joint classification problem for k binary classes, and we train on the surrogate loss of cross entropy to make the objective differentiable. While classical optimal codes such as Turbo, LDPC, and Polar codes all have *linear* encoders, appropriately parametrizing both the encoder $g_\theta(\cdot)$ and the decoder $f_\phi(\cdot)$ by neural networks (NN) allows for a much broader class of codes, especially non-linear codes. However, in the absence of any structure, NNs fail to learn non-trivial codes and end up performing worse than simply repeating each message bit n/k times [8, 9].

A fundamental question in machine learning for channel coding is thus: how do we design architectures for our neural encoders and decoders that give the appropriate inductive bias? To gain intuition towards addressing this, we focus on Reed-Muller (RM) codes. In §4.3, we present a novel family of non-linear codes, *KO codes*, that strictly generalize and improve upon RM codes by capitalizing on their inherent recursive structure. Our approach seamlessly generalizes to Polar codes, explained in §4.5.

4.2.2 Reed-Muller (RM) codes

We use a small example of RM(3, 1) and refer to Appendix C.5 for the larger example in our main results.

Encoding. RM codes are a family of codes parametrized by a variable size $m \in \mathbb{Z}_+$ and an order $r \in \mathbb{Z}_+$ with $r \leq m$, denoted as RM(m, r). It is defined by an *encoder*, which maps binary information bits $\boldsymbol{m} \in \{0, 1\}^k$ to codewords $\boldsymbol{x} \in \{0, 1\}^n$. RM$(m, r)$ code sends $k = \sum_{i=0}^{r} \binom{m}{i}$ information bits with $n = 2^m$ transmissions. The *code distance* measures the minimum distance between all (pairs of) codewords. Table 4.1 summarizes these parameters.

Code length	Code dimension	Rate	Distance
$n = 2^m$	$k = \sum_{i=0}^{r} \binom{m}{i}$	$\rho = k/n$	$d = 2^{m-r}$

Table 4.1: Parameters of a RM(m, r) code

One way to define RM(m, r) code is via the recursive application of a *Plotkin construction*. The basic building block is a mapping Plotkin : $\{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^{2\ell}$, where

$$\text{Plotkin}(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v}) \ , \tag{4.2}$$

with $\oplus$ representing a coordinate-wise XOR and $(\cdot, \cdot)$ denoting concatenation of two vectors [50].

In view of the Plotkin construction, RM codes are recursively defined as a set of codewords of the form:

$$\begin{aligned} \text{RM}(m, r) = \{(\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v}) : \boldsymbol{u} \in \text{RM}(m-1, r), \\ \boldsymbol{v} \in \text{RM}(m-1, r-1)\}, \end{aligned} \tag{4.3}$$

where RM$(m, 0)$ is a repetition code that repeats a single information bit 2^m times, i.e., $\boldsymbol{x} = (m_1, m_1, \ldots, m_1)$. When $r = m$, the full-rate RM(m, m) code is also recursively defined as a Plotkin construction of two RM$(m-1, m-1)$ codes. Unrolling the recursion in Eq. (4.3), a RM(m, r) encoder can be represented by a corresponding (rooted and binary) computation tree, which we refer to as its *Plotkin tree*. In this tree, each branch represents a Plotkin mapping of two codes of appropriate lengths, recursively applied from the leaves to the root.

Figure 4.3a illustrates such a Plotkin tree decomposition of RM$(3, 1)$ encoder. Encoding starts from the bottom right leaves. The leaf RM$(1, 0)$ maps m_3 to (m_3, m_3) (repetition), and another leaf RM$(1, 1)$ maps (m_1, m_2) to $(m_1, m_1 \oplus m_2)$ (Plotkin mapping of two RM$(0, 0)$ codes). Each branch in this tree performs the Plotkin construction of Eq. (4.2). The next operation is the parent of these two leaves which performs Plotkin(RM$(1, 1)$, RM$(1, 0)$) $=$ Plotkin$((m_1, m_1 \oplus m_2), (m_3, m_3))$ which outputs the vector $(m_1, m_1 \oplus m_2, m_1 \oplus m_3, m_1 \oplus m_2 \oplus m_3)$, which is known as RM$(2, 1)$ code. This coordinate-wise Plotkin construction is applied recursively one more time to combine

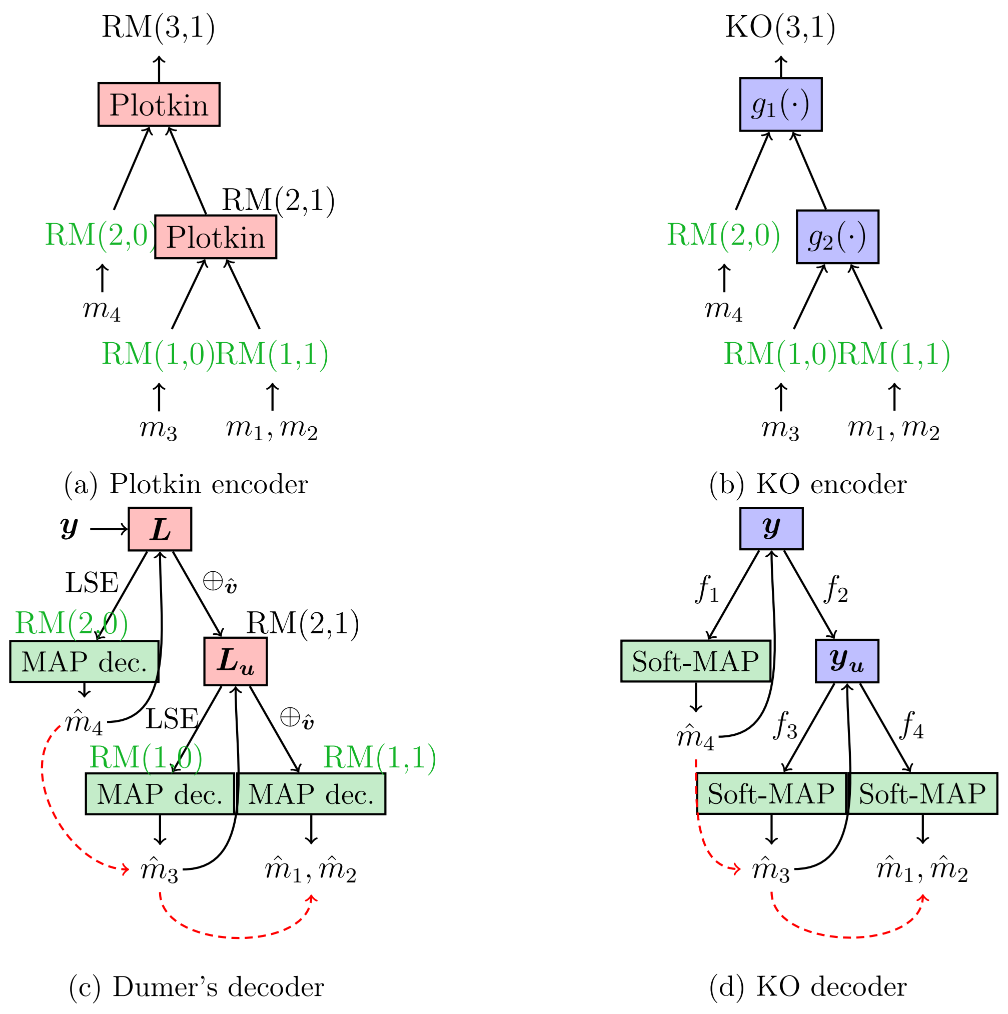


Figure 4.3: Plotkin trees for RM$(3, 1)$ and KO$(3, 1)$ codes. Leaves are shown in green. Red arrows indicate the bit decoding order.

RM$(2, 0)$ and RM$(2, 1)$ at the root of the tree. The resulting codewords are RM$(3, 1)$ = Plotkin(RM$(2, 1)$, RM$(2, 0)$) = Plotkin$((m_1, m_1 \oplus m_2, m_1 \oplus m_3, m_1 \oplus m_2 \oplus m_3), (m_4, m_4, m_4, m_4))$.

This recursive structure of RM codes (i) inherits the good minimum distance property of the Plotkin construction and (ii) enables efficient decoding.

Decoding. Since the classical Reed's algorithm [55], there have been several decoders for RM codes; [56] is a detailed survey. We focus on the most efficient one, called *Dumer's recursive decoding* [1, 57, 58] that fully capitalizes on the recursive Plotkin construction in Eq. (4.3). The basic principle is: to decode an RM codeword $\boldsymbol{x} = (\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v}) \in \text{RM}(m, r)$, we first recursively decode the left sub-codeword $\boldsymbol{v} \in \text{RM}(m-1, r-1)$, and then the right sub-codeword $\boldsymbol{u} \in \mathrm{RM}(m-1, r)$, and we use them together to stitch back the original codeword. This recursion is continued until we reach the leaf nodes where we perform maximum a posteriori (MAP) decoding. Dumer's recursive decoding is also referred to as *successive cancellation* decoding in the context of polar codes [59].

Figure 4.3c illustrates this decoding procedure for $\mathrm{RM}(3,1)$. Dumer's decoding starts at the root and uses the soft-information of codewords to decode the message bits. Suppose that the message bits $\boldsymbol{m} = (m_1, \ldots, m_4)$ are encoded into an $\mathrm{RM}(3,1)$ codeword $\boldsymbol{x} \in \{0,1\}^8$ using the Plotkin encoder in Figure 4.3a. Let $\boldsymbol{y} \in \mathbb{R}^8$ be the corresponding noisy codeword received at the decoder. To decode the bits $\boldsymbol{m}$, we first obtain the soft-information of the codeword $\boldsymbol{x}$, i.e., we compute its Log-Likehood-Ratio (LLR) $\boldsymbol{L} \in \mathbb{R}^8$:

$$L_i = \log \frac{\mathbb{P}\left(y_i | x_i = 0\right)}{\mathbb{P}\left(y_i | x_i = 1\right)}, \quad i = 1, \ldots, 8.$$

We next use $\boldsymbol{L}$ to compute soft-information for its left and right children: the $\mathrm{RM}(2,0)$ codeword $\boldsymbol{v}$ and the $\mathrm{RM}(2,1)$ codeword $\boldsymbol{u}$. We start with the left child, $\boldsymbol{v}$.

Since the codeword $\boldsymbol{x} = (\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v})$, we can also represent its left child as $\boldsymbol{v} = \boldsymbol{u} \oplus (\boldsymbol{u} \oplus \boldsymbol{v}) = \boldsymbol{x}_{1:4} \oplus \boldsymbol{x}_{5:8}$. Hence its LLR vector $\boldsymbol{L}_{\boldsymbol{v}} \in \mathbb{R}^4$ can be readily obtained from that of $\boldsymbol{x}$. In particular it is given by the log-sum-exponential transformation: $\boldsymbol{L}_{\boldsymbol{v}} = \mathrm{LSE}(\boldsymbol{L}_{1:4}, \boldsymbol{L}_{5:8})$, where $\mathrm{LSE}(a,b) \triangleq \log((1+e^{a+b})/(e^a + e^b))$ for $a, b \in \mathbb{R}$. Since this feature $\boldsymbol{L}_{\boldsymbol{v}}$ corresponds to a repetition code, $\boldsymbol{v} = (m_4, m_4, m_4, m_4)$, majority decoding (same as the MAP) on the sign of $\boldsymbol{L}_{\boldsymbol{v}}$ yields the decoded message bit as $\hat{m}_4$. Finally, the left codeword is decoded as $\hat{\boldsymbol{v}} = (\hat{m}_4, \hat{m}_4, \hat{m}_4, \hat{m}_4)$.

Having decoded the left $\mathrm{RM}(2,0)$ codeword $\hat{\boldsymbol{v}}$, our goal is to now obtain soft-information $\boldsymbol{L}_{\boldsymbol{u}} \in \mathbb{R}^4$ for the right $\mathrm{RM}(2,1)$ codeword $\boldsymbol{u}$. Fixing $\boldsymbol{v} = \hat{\boldsymbol{v}}$, notice that the codeword $\boldsymbol{x} = (\boldsymbol{u}, \boldsymbol{u} \oplus \hat{\boldsymbol{v}})$ can be viewed as a 2-repetition of $\boldsymbol{u}$ depending on the parity of $\hat{\boldsymbol{v}}$. Thus the LLR $\boldsymbol{L}_{\boldsymbol{u}}$ is given by LLR addition accounting for the parity of $\hat{\boldsymbol{v}}$: $\boldsymbol{L}_{\boldsymbol{u}} = \boldsymbol{L}_{1:4} \oplus_{\hat{\boldsymbol{v}}} \boldsymbol{L}_{5:8} = \boldsymbol{L}_{1:4} + (-1)^{\hat{\boldsymbol{v}}} \boldsymbol{L}_{5:8}$. Since $\mathrm{RM}(2,1)$ is an internal node in the tree, we again recursively decode its left child $\mathrm{RM}(1,0)$ and its right child $\mathrm{RM}(1,1)$ which are both leaves. For $\mathrm{RM}(1,0)$, decoding is similar to that of $\mathrm{RM}(2,0)$ above, and we obtain its information bit $\hat{m}_3$ by first applying the log-sum-exponential function on the feature $\boldsymbol{L}_{\boldsymbol{u}}$ and then majority decoding. Likewise, we obtain the LLR feature

$\boldsymbol{L_{uu}} \in \mathbb{R}^2$ for the right RM$(1,1)$ child using parity-adjusted LLR addition on $\boldsymbol{L_u}$. Finally, we decode its corresponding bits $(\hat{m}_1, \hat{m}_2)$ using efficient MAP-decoding of first order RM codes [56]. Thus we obtain the full block of decoded message bits as $\hat{\boldsymbol{m}} = (\hat{m}_1, \hat{m}_2, \hat{m}_3, \hat{m}_4)$.

An important observation from Dumer's algorithm is that the sequence of bit decoding in the tree is: RM$(2,0) \to$ RM$(1,0) \to$ RM$(1,1)$. A similar decoding order holds for all RM$(m,2)$ codes, where all the left leaves (order-1 codes) are decoded first from top to bottom, and the right-most leaf (full-rate RM$(2,2)$) is decoded at the end.

4.3 KO codes: Novel Neural codes

We design KO codes using the Plotkin tree as the skeleton of a new neural network architecture which strictly improve upon their classical counterparts. **KO encoder**. Earlier we saw the design of RM codes via recursive Plotkin mapping. Inspired by this elegant construction, we present a new family of codes, called *KO codes*, denoted as KO(m, r, g_θ, f_ϕ). These codes are parametrized by a set of four parameters: a non-negative integer pair (m, r), a finite set of encoder neural networks g_θ, and a finite set of decoder neural networks f_ϕ. In particular, for any fixed pair (m, r), our KO encoder inherits the same code parameters (k, n, ρ) and the same Plotkin tree skeleton of the RM encoder. However, a critical distinguishing component of our KO(m, r) encoder is a set of encoding neural networks $g_\theta = \{g_i\}$ that strictly generalize the Plotkin mapping; to each internal node i of the Plotkin tree, we associate a neural network g_i that applies a coordinate-wise real valued non-linear mapping $(\boldsymbol{u}, \boldsymbol{v}) \mapsto g_i(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{R}^{2\ell}$ as opposed to the classical binary valued Plotkin mapping $(\boldsymbol{u}, \boldsymbol{v}) \mapsto (\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v}) \in \{0,1\}^{2\ell}$. Figure 4.3b illustrates this for the KO$(3,1)$ encoder.

The significance of our KO encoder g_θ is that by allowing for general nonlinearities g_i to be learnt at each node, we enable for a much richer and broader class of nonlinear encoders and codes to be discovered on a whole which contribute to non-trivial gains over standard RM codes. Further, we have the same encoding complexity as that of an RM encoder since each $g_i : \mathbb{R}^2 \to \mathbb{R}$ is applied coordinate-wise on its vector inputs. The parameters of these neural networks g_i are trained via stochastic gradient descent on the cross entropy loss. See §C.7 for experimental details.

KO decoder. Training the encoder is possible only if we have a corresponding decoder. This necessitates the need for an efficient family of matching decoders. Inspired by the Dumer's decoder, we present a new family of *KO decoders* that fully capitalize on the recursive structure of KO encoders via the Plotkin tree.

Our KO decoder has three distinct features:

(i) Neural decoder: The KO decoder architecture is parametrized by a set of decoding neural networks $f_\phi = \{(f_{2i-1}, f_{2i})\}$. Specifically, to each internal node i in the tree, we associate f_{2i-1} to its left branch whereas f_{2i} corresponds to the right branch. Figure 4.3d shows this for the KO$(3,1)$ decoder. The pair of decoding neural networks (f_{2i-1}, f_{2i}) can be viewed as matching decoders for the corresponding encoding network g_i: While g_i encodes the left and right codewords arriving at this node, the outputs of f_{2i-1} and f_{2i} represent appropriate Euclidean feature vectors for decoding them. Further, f_{2i-1} and f_{2i} can also be viewed as a generalization of Dumer's decoding to nonlinear real codewords: f_{2i-1} generalizes the LSE function, while f_{2i} extends the operation $\oplus_{\hat{\boldsymbol{v}}}$. Note that both the functions f_{2i-1} and f_{2i} are also applied coordinate-wise and hence we inherit the same decoding complexity as Dumer's.

(ii) Soft-MAP decoding: Since the classical MAP decoding to decode the bits at the leaves is not differentiable, we design a new differentiable counterpart, the *Soft-MAP decoder*. Soft-MAP decoder enables gradients to pass through it which is crucial for training the neural (encoder, decoder) pair (g_θ, f_ϕ) in an end-to-end manner.

(iii) Channel agnostic: Our decoder directly operates on the received noisy codeword $\boldsymbol{y} \in \mathbb{R}^n$ while Dumer's decoder uses its LLR transformation $\boldsymbol{L} \in \mathbb{R}^n$. Thus, our decoder can learn the appropriate channel statistics for decoding directly from $\boldsymbol{y}$ alone; in contrast, Dumer's algorithm requires precise channel characterization which is not usually known.

4.4 Main results

We train the KO encoder g_θ and KO decoder f_ϕ from §4.3 using an approximation of the BER loss in (4.1). The details are provided in §C.7. In this section we focus on the second-order KO$(8,2)$ and KO$(9,2)$ codes.

4.4.1 KO codes improve over RM codes

In Figure 4.1, the trained KO$(9,2)$ improves over the competing RM$(9,2)$ both in BER and BLER. The superiority in BLER is unexpected as our training loss is a surrogate for the BER. Though one would prefer to train on BLER as it is more relevant in practice, it is challenging to design a surrogate loss for BLER that is also differentiable: all literature on learning decoders minimize only BER [60–62]. Consequently, improvements in BLER with trained encoders and/or decoders are rare. We discover a code that improves both BER and BLER, and we observe a similar gain with KO$(8,2)$ in Figure 4.4. Performance of a binarized version KO-b$(8,2)$ is also shown which we describe further in §4.4.4.

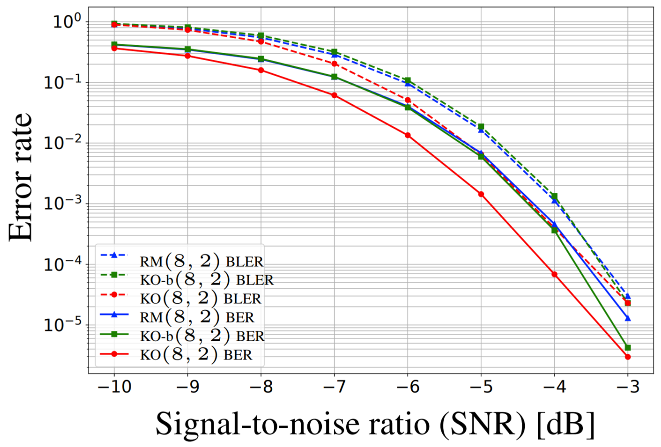


Figure 4.4: Neural network based KO$(8,2)$ and KO-b$(8,2)$ improve upon RM$(8,2)$ in BER and BLER, but the gain is small for the binarized codewords of KO-b$(8,2)$ (for all the codes, the code dimension is 37 and block length is 256).

4.4.2 Interpreting KO codes

We interpret the learned encoders and decoders to explain the source of the performance gain.

Interpreting the KO encoder. To interpret the learned KO code, we examine the pairwise distance between codewords. In classical linear coding, pairwise distances are expressed in terms of the weight distribution of the code, which counts how many codewords of each specific Hamming weight $1, 2, \ldots, n$ exist in the code. The weight distribution of linear codes are used to derive analytical bounds that can be explicitly computed on the BER and BLER over AWGN channels [63]. For nonlinear codes, however, the weight distribution does not capture pairwise distances. Therefore, we explore the distribution of all the pairwise distances of non-linear KO codes that can play the same role as the weight distribution does for linear codes.

The pairwise distance distribution of the RM codes remains an active area of research as it is used to prove that RM codes achieve the capacity [64–66] (Figure 4.5 blue). However, these results are asymptotic in the block length and do not guarantee a good performance, especially in the small-to-medium block lengths that we are interested in. On the other hand, Gaussian codebooks, codebooks randomly picked from the ensemble of all Gaussian codebooks, are known to be asymptotically optimal, i.e., achieving the capacity [2], and also demonstrate optimal finite-length scaling laws closely related to the pairwise distance distribution [67] (Figure 4.5 orange).

Remarkably, the pairwise distance distribution of KO code shows a staggering resemblance to that of the Gaussian codebook of the same rate ρ and blocklength n (Figure 4.5 red). This is an unexpected phenomenon since we minimize only BER. We posit that the NN training has learned to construct a Gaussian-like codebook in order to minimize BER. Most importantly, unlike the Gaussian codebook, KO codes constructed via NN training are fully compatible with efficient decoding. This phenomenon is observed for all order-2 codes we trained (e.g., Figure 4.2 for KO$(9, 2)$).

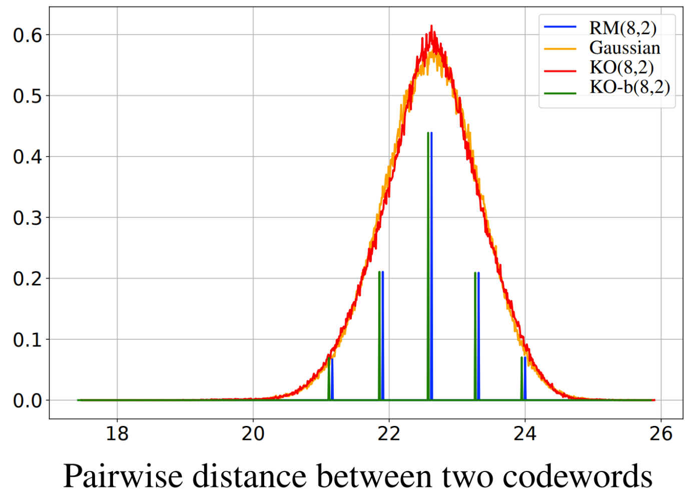


Figure 4.5: Histograms of pairwise distances between codewords for $(8,2)$ codes reveal that $\mathrm{KO}(8,2)$ code has learned an approximate Gaussian codebook that can be efficiently decoded.

Interpreting the KO decoder. We now analyze how the KO decoder contributes to the gains in BLER over the RM decoder. Let $\boldsymbol{m} = (\boldsymbol{m}_{(7,1)}, \ldots, \boldsymbol{m}_{(2,2)})$ denote the block of transmitted message bits where the ordered set of indices $\mathcal{L} = \{(7,1), \ldots, (2,2)\}$ correspond to the leaf branches (RM codes) of the Plotkin tree. Let $\hat{\boldsymbol{m}}$ be the decoded estimate by the $\mathrm{KO}(8,2)$ decoder.

We provide Plotkin trees of $\mathrm{RM}(8,2)$ and $\mathrm{KO}(8,2)$ decoders in Figures C.5a and C.5b in the appendix. Recall that for this $\mathrm{KO}(8,2)$ decoder, similar to the $\mathrm{KO}(3,1)$ decoder in Figure 4.3d, we decode each sub-code in the leaves sequentially, starting from the $(7,1)$ branch down to $(2,2)$: $\hat{\boldsymbol{m}}_{(7,1)} \to \ldots \to \hat{\boldsymbol{m}}_{(2,2)}$. In view of this decoding order, BLER, defined as $\mathbb{P}\left(\hat{\boldsymbol{m}} \neq \boldsymbol{m}\right)$, can be decomposed as

$$\mathbb{P}\left(\hat{\boldsymbol{m}} \neq \boldsymbol{m}\right) = \sum_{i \in \mathcal{L}} \mathbb{P}\left(\hat{\boldsymbol{m}}_i \neq \boldsymbol{m}_i, \hat{\boldsymbol{m}}_{1:i-1} = \boldsymbol{m}_{1:i-1}\right). \tag{4.4}$$

In other words, BLER can also be represented as the sum of the fraction of errors the decoder makes in each of the leaf branches when no errors were made in the previous ones. Thus, each term in Eq. (4.4) can be viewed as the contribution of each sub-code to the total BLER.

This is plotted in Figure 4.6 which shows that the $\mathrm{KO}(8,2)$ decoder achieves better BLER than the $\mathrm{RM}(8,2)$ decoder by making major gains in the leftmost $(7, 1)$ branch (which is decoded first) at the expense of other branches. However, the decoder (together with the encoder) has learnt to better balance these contributions evenly across all branches, resulting in lower BLER overall. The unequal errors in the branches of the RM code has been observed before, and some efforts have been made to balance them [68]; the fact KO codes learn such a balancing scheme purely from data is, perhaps, remarkable.

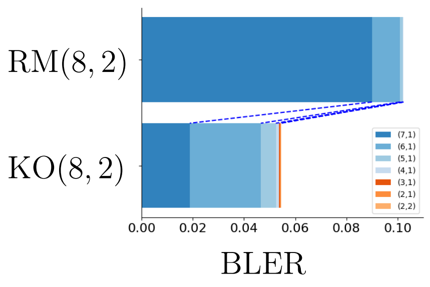


Figure 4.6: Separating each sub-code contribution in the KO$(8, 2)$ decoder and the RM$(8, 2)$ decoder reveals that KO$(8, 2)$ improves in the total BLER by balancing the contributions more evenly over the sub-codes.

4.4.3 Robustness to non-AWGN channels

As the environment changes dynamically in real world channels, robustness is crucial in practice. We therefore test the KO code under canonical channel models and demonstrate robustness, i.e., the ability of a code trained on AWGN to perform well under a different channel *without retraining*. It is well known that Gaussian noise is the worst case noise among all noise with the same variance [2, 69] when an optimal decoder is used which might take an exponential time. When decoded with efficient decoders, as we do with both RM and KO codes, catastrophic failures have been reported in the case of Turbo decoders [8]. We show that both RM codes and KO codes are robust and that KO codes maintain their gains over RM codes as the channels vary.

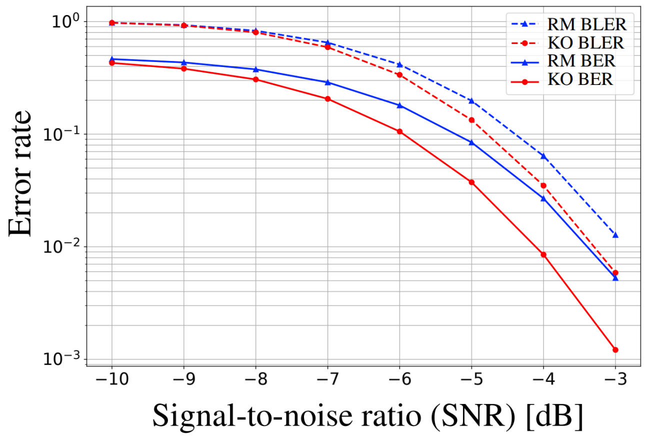


Figure 4.7: KO$(8, 2)$ trained on AWGN is robust when tested on a fast fading channel and maintains a significant gain over RM(8,2).

We first test on a *Rayleigh fast fading channel*, defined as $y_i = a_i x_i + n_i$, where x_i is the transmitted symbol, y_i is the received symbol, $n_i \sim \mathcal{N}(0, \sigma^2)$ is the additive Gaussian noise, and a is from a Rayleigh distribution with the variance of a chosen as $\mathbb{E}[a_i^2] = 1$.

We next test on a bursty channel, defined as $y_i = x_i + n_i + w_i$, where x_i is the input symbol, y_i is the received symbol, $n_i \sim \mathcal{N}(0, \sigma^2)$ is the additive Gaussian noise, and $w_i \sim \mathcal{N}(0, \sigma_b^2)$ with probability ρ and $w_i = 0$ with probability $1 - \rho$. In the experiment, we choose $\rho = 0.1$ and $\sigma_b = \sqrt{2}\sigma$.

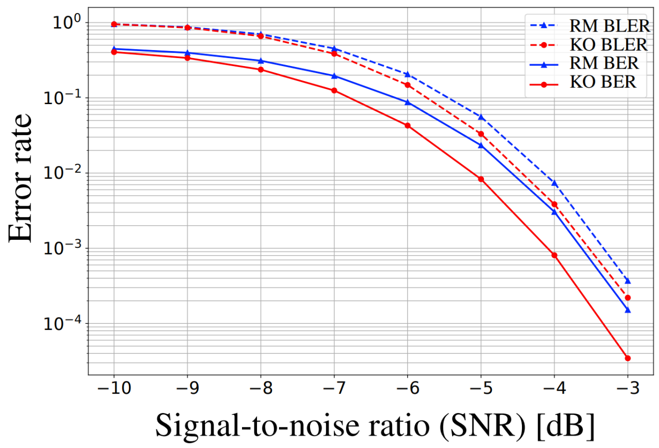


Figure 4.8: KO$(8, 2)$ trained on AWGN is robust when tested on a bursty channel and maintains a significant gain over RM$(8, 2)$.

4.4.4 Ablation studies

In comparison to the classical RM codes, the KO codes have two additional features: real-valued codewords and non-linearity. It is thus natural to ask how each of these components contribute to its gains over RM codes. To evaluate their contribution, we did ablation experiments for KO$(8, 2)$: (i) First, we constrain the KO codewords to be binary but allow for non-linearity in the encoder g_θ. The performance of this binarized version KO-b$(8, 2)$ is illustrated in Figure 4.9 below. We observe that this binarized KO-b$(8, 2)$ performs similar to RM$(8, 2)$ except for slight gains at high SNRs but uniformly worse than KO$(8, 2)$. (ii) Now we transmit the real-valued codewords but constrain the encoder g_θ to be linear (in real-value operations). The resulting code, KO-linear$(8, 2)$, performs almost identical to RM$(8, 2)$ but worse than KO$(8, 2)$, as highlighted by the orange curve in Figure 4.9.

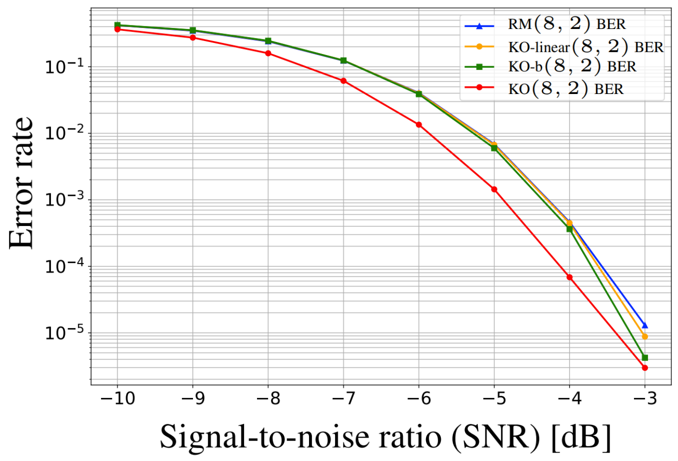


Figure 4.9: Ablation studies highlight that both non-linearity and real-valued codewords are equally important for good performance of KO codes. The linear version, KO-linear$(8, 2)$, and the binary version, KO-b$(8, 2)$, both perform worse than KO$(8, 2)$ and similar to RM$(8, 2)$.

These ablation experiments suggest us that presence of both the non-linearity and real-valued codewords are necessary for the good performance of KO codes and removal of any of these components hurts the gains it achieves over RM codes. Further, this also highlights that in absence of either of these components, the performance drops back to that of the original RM codes.

4.4.5 Complexity of KO decoding

Ultra-Reliable Low Latency Communication (URLLC) is increasingly required for modern applications including vehicular communication, virtual reality, and remote robotics [70, 71]. In general, a KO(m, r) code requires $O(n \log n)$ operations to decode which is the same as the efficient Dumer's decoder for an RM(m, r) code, where $n = 2^m$ is the block length. More precisely, the successive cancellation decoder for RM$(8, 2)$ requires 11268 operations whereas KO$(8, 2)$ requires 550644 operations which we did not try to optimize for this project. We discuss promising preliminary results in reducing the computational complexity in §4.6, where KO decoders achieve a computational efficiency comparable to the successive cancellation decoders of RM codes.

4.5 KO codes improve upon Polar codes

Results from §4.4 demonstrate that our KO codes significantly improve upon RM codes on a variety of benchmarks. Here, we focus on a different family of capacity-achieving landmark codes: *Polar codes* [59].

Polar and RM codes are closely related, especially from an encoding point-of-view. The generator matrices of both codes are chosen from the same parent square matrix by following different row selection rules. More precisely, consider a RM(m, r) code that has code dimension $k = \sum_{i=0}^{r} \binom{m}{i}$ and block-length $n = 2^m$. Its encoding generator matrix is obtained by picking the k rows of the square matrix $\boldsymbol{G}_{n\times n} := \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{\otimes m}$ that have the largest Hamming weights (i.e., Hamming weight of at least 2^{m-r}), where $[\cdot]^{\otimes m}$ denotes the m-th Kronecker power. The Polar encoder, on the other hand, picks the rows of $\boldsymbol{G}_{n\times n}$ that correspond to the most reliable bit-channels [59].

The recursive Kronecker structure inherent to the parent matrix $\boldsymbol{G}_{n\times n}$ can also be represented by a computation graph, a complete binary tree. Thus the corresponding computation tree for a Polar code is obtained by freezing a set of leaves (row-selection). We refer to this encoding computation graph of a Polar code as its *Plotkin tree*. This Plotkin tree structure of Polar codes enables a matching efficient decoder: the *successive cancellation* (SC). The SC decoding algorithm is similar to Dumer's decoding for RM codes. Hence, Polar codes can be completely characterized by their corresponding Plotkin trees.

Inspired by the Kronecker structure of Polar Plotkin trees, we design a new family of KO codes to strictly improve upon them. We build a novel NN architecture that capitalizes on the Plotkin tree skeleton and generalizes it to nonlinear codes. This enables us to discover new nonlinear algebraic structures. The KO encoder and decoder can be trained in an end-to-end manner using variants of stochastic gradient descent (§C.1).

In Figure 4.10, we compare the performance of our KO code with its competing Polar$(64, 7)$ code, i.e., code dimension $k = 7$ and block length $n = 64$, in terms of BER. Figure 4.10 highlights that our KO code achieves significant gains over Polar$(64, 7)$ on a wide range of SNRs. In particular, we obtain a gain of almost 0.7 dB compared to that of Polar at the BER 10^{-4}. For comparison we also plot the performance of both codes with the optimal MAP decoding. We observe that the BER curve of our KO decoder, unlike the SC decoder, almost matches that of the MAP decoder, convincingly demonstrating its optimality.

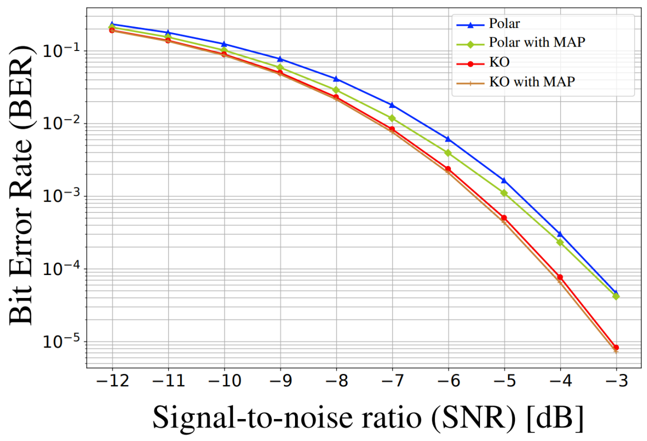


Figure 4.10: Neural network based KO code improves upon the Polar$(64, 7)$ code when trained on AWGN channel. KO decoder also matches the optimal MAP decoder.

We also observe similar improvements for BLER (Figure C.2, §C.1). This successful case study with training KO (encoder, decoder) pairs further demonstrates that our novel neural architectures seamlessly generalize to codes with an underlying Kronecker product structure.

4.6 Tiny KO

In this section, we focus on further reducing the total number of mathematical operations required for our KO decoder with the objective of achieving similar computational efficiency as the successive cancellation decoder of RM codes.

As detailed in §C.7.2, each neural component in the KO encoder and decoder has 3 hidden layers with 32 nodes each. For the decoder, the total number of parameters in each decoder neural block is 69×32. We replace all neural blocks with a smaller one with one hidden layer of four nodes. This decoder neural block has 20 parameters, obtaining a factor of 110 compression in the number of parameters. The computational complexity of this compressed decoder, which we refer to as TinyKO, is within a factor of four from Dumer's successive cancellation decoder. Each neural network component has two matrix multiplication steps and one activation function on a vector which can be fully parallelized on a GPU. With the GPU parallelization, TinyKO has the same time complexity/latency as Dumer's SC decoding.

Table 4.2 shows that there is almost no loss in reliability for the compressed KO$(8,2)$ encoder and decoder in this manner. Training a smaller neural network take about two times more iterations compared to the larger one, although each iteration is faster for the smaller network.

SNR (dB)	TinyKO$(8,2)$ BER	KO$(8,2)$ BER
-10	0.38414 ± 2e-7	0.36555 ± 2e-7
-9	0.29671 ± 2e-7	0.27428 ± 2e-7
-8	0.18037 ± 2e-7	0.15890 ± 2e-7
-7	0.07455 ± 2e-7	0.06167 ± 1e-7
-6	0.01797 ± 8e-8	0.01349 ± 7e-8
-5	2.18083e-3 ± 3e-8	1.46003e-3 ± 2e-8
-4	1.18919e-4 ± 7e-9	0.64702e-4 ± 4e-9
-3	4.54054e-6 ± 1e-9	3.16216e-6 ± 1e-9

Table 4.2: The smaller TinyKO neural architecture with 100 times smaller number of parameters achieve similar bit-error-rates as the bigger KO architecture.

If one is allowed more computation time (e.g., $O(n^r \log n)$), then [53] proposes a recursive projection-aggregation (RPA) decoder for RM(m, r) codes that significantly improves over Dumer's successive cancellation. With list decoding, this is empirically shown to approach the performance of the MAP decoder. It is a promising direction to explore deep learning architectures upon the computation tree of the RPA decoders to design new family of codes.

Chapter 5

Conclusion

In this dissertation, we gave initial evidence, in the form of KO codes, to our conjecture that deep-learning can play a crucial role in the discovery of codes and hence in addressing the long standing goals of coding theory. While these results are indeed an encouraging first step, there are many exciting future research directions along this line. We outline them below.

Codes for 5G and beyond

While KO codes make some initial progress along this line in the context of point-to-point communication, it would be interesting to build upon these results to construct codes for more complicated multi-terminal settings: the interference channel, the relay channel, and the feedback channel. Designing optimal codes in these scenarios is a long standing open research problem. Leveraging DL tools to address this would be an exciting step. Apart from communication, designing codes for distributed computation, data compression, and data storage is also a fruitful avenue.

Inventing new coding structures

A common theme in using DL to design codes is to rely on classical linear codes and use their structure as an anchor to construct their non-linear counterparts, aka the neural augmentation principle. For example, TurboAE [9] uses the sequential aspect of convolutional/turbo codes whereas our KO codes capitalize on the recursive tree property of RM/polar codes. However, such an approach is limited by the existing code structures which are in turn limited by human ingenuity. An interesting open question here is to automate the design of these structures and discover a broad family of them directly from data.

Theory of DL via information/coding-theoretic lens

Over the past few years, motivated by the empirical success of DL, there has been a lot of interest in DL theory. However, to the best of our knowledge, most of these theoretical works are driven by the DL phenomena observed in computer vision (CV), natural language processing (NLP), and reinforcement learning. For example, the recent interest in the analysis of over-parameterized neural networks (NNs) is inspired by the fact that these models generalize well in CV and NLP applications, despite being heavily over-parameterized [72]. In view of this, buoyed by the added success of DL in the communication domain, we would like to propose the information/coding-theoretic framework of end-to-end communication as a new guiding lens to understand DL models. Some interesting topics along this line include analysis of the class of codes learnt by NNs, the loss landscape of NNs, and their generalization properties, etc. A key leverage of this new framework compared to CV or NLP is that the underlying communication model and the data generative process are mathematically well defined. Hence we can capitalize on this feature to obtain insights about DL models in a principled manner. Further, this synergy between communication and DL can yield algorithmic insights for designing the next generation codes.

References

[1] I. Dumer and K. Shabunov, "Soft-decision decoding of reed-muller codes: recursive lists," *IEEE Transactions on information theory*, vol. 52, no. 3, pp. 1260–1266, 2006.

[2] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[3] T. Richardson and R. Urbanke, *Modern coding theory.* Cambridge University Press, 2008.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton et al., "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[8] H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, and P. Viswanath, "Communication algorithms via deep learning," *arXiv preprint arXiv:1805.09317*, 2018.

[9] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, "Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels," in *Advances in Neural Information Processing Systems*, 2019, pp. 2758–2768.

[10] M. Welling, "Neural augmentation in wireless communication," 2020.

[11] V. Tresp, "Mixtures of gaussian processes," ser. NIPS, 2001.

[12] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of SVMs for very large scale problems," *Neural Computing*, 2002.

[13] J. W. Ng and M. P. Deisenroth, "Hierarchical mixture-of-experts model for large-scale gaussian process regression," *arXiv preprint arXiv:1412.3078*, 2014.

[14] L. Theis and M. Bethge, "Generative image modeling using spatial lstms," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 1927–1935.

[15] P. Le, M. Dymetman, and J.-M. Renders, "Lstm-based mixture-of-experts for knowledge-aware dialogues," *arXiv preprint arXiv:1605.01652*, 2016.

[16] S. Gross, A. Szlam et al., "Hard mixtures of experts for large scale weakly supervised vision," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on.* IEEE, 2017, pp. 5085–5093.

[17] X. Sun, X. Peng, F. Ren, and Y. Xue, "Human-machine conversation based on hybrid neural network," in *Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), 2017 IEEE International Conference on*, vol. 1. IEEE, 2017, pp. 260–266.

[18] X. Wang, F. Yu, R. Wang, Y.-A. Ma, A. Mirhoseini, T. Darrell, and J. E. Gonzalez, "Deep mixture of experts via shallow embedding," *arXiv preprint arXiv:1806.01531*, 2018.

[19] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, 1991.

[20] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," vol. abs/1412.3555, 2014.

[21] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[23] H. Sedghi, M. Janzamin, and A. Anandkumar, "Provable tensor methods for learning mixtures of classifiers," *arXiv preprint arXiv:1412.3046*, 2014.

[24] Y. Sun, S. Ioannidis, and A. Montanari, "Learning mixtures of linear classifiers," in *Proceedings of the 31st International Conference on Machine Learning*, vol. 32, no. 2, 2014, pp. 721–729.

[25] X. Yi, C. Caramanis, and S. Sanghavi, "Solving a mixture of many random linear equations by tensor decomposition and alternating minimization," *arXiv preprint arXiv:1608.05749*, 2016.

[26] K. Zhong, P. Jain, and I. S. Dhillon, "Mixed linear regression with multiple components," 2016, pp. 2190–2198.

[27] S. Balakrishnan, M. J. Wainwright, and B. Yu, "Statistical guarantees for the EM algorithm: From population to sample-based analysis," *The Annals of Statistics*, vol. 45, no. 1, pp. 77–120, 2017.

[28] J. B. Kruskal, "Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear algebra and its applications*, vol. 18, no. 2, pp. 95–138, 1977.

[29] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[30] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2773–2832, Jan. 2014.

[31] M. Janzamin, H. Sedghi, and A. Anandkumar, "Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods," *arXiv preprint arXiv:1506.08473*, 2015.

[32] Y. Li and Y. Yuan, "Convergence analysis of two-layer neural networks with relu activation," in *Advances in Neural Information Processing Systems*, 2017, pp. 597–607.

[33] R. Ge, J. D. Lee, and T. Ma, "Learning one-hidden-layer neural networks with landscape design," *arXiv preprint arXiv:1711.00501*, 2017.

[34] K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon, "Recovery guarantees for one-hidden-layer neural networks," *arXiv preprint arXiv:1706.03175*, 2017.

[35] S. S. Du, J. D. Lee, Y. Tian, B. Poczos, and A. Singh, "Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima," *arXiv preprint arXiv:1712.00779*, 2017.

[36] I. Safran and O. Shamir, "Spurious local minima are common in two-layer relu neural networks," *arXiv preprint arXiv:1712.08968*, 2017.

[37] G. McLachlan and T. Krishnan, *The EM algorithm and extensions*. John Wiley & Sons, 2007, vol. 382.

[38] V. Sharan and G. Valiant, "Orthogonalized ALS: A theoretically principled tensor decomposition algorithm for practical use," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, 06–11 Aug 2017. [Online]. Available: http://proceedings.mlr.press/v70/sharan17a.html pp. 3095–3104.

[39] M. Janzamin, H. Sedghi, and A. Anandkumar, "Score function features for discriminative learning: Matrix and tensor framework," vol. abs/1412.2863, 2014. [Online]. Available: http://arxiv.org/abs/1412.2863

[40] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Advances in neural information processing systems*, 2014, pp. 855–863.

[41] R. Ge, J. D. Lee, and T. Ma, "Learning one-hidden-layer neural networks with landscape design," vol. abs/1711.00501, 2017. [Online]. Available: http://arxiv.org/abs/1711.00501

[42] W. Gao, A. V. Makkuva, S. Oh, and P. Viswanath, "Learning one-hidden-layer neural networks under general input distributions," in *Proceedings of Machine Learning Research*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and M. Sugiyama, Eds., vol. 89. PMLR, 16–18 Apr 2019. [Online]. Available: http://proceedings.mlr.press/v89/gao19b.html pp. 1950–1959.

[43] A. V. Makkuva, S. Oh, S. Kannan, and P. Viswanath, "Breaking the gridlock in mixture-of-experts: Consistent and efficient algorithms," *International Conference on Machine Learning (ICML 2019), arXiv preprint arXiv:1802.07417*, 2019.

[44] H. Grad, "Note on n-dimensional hermite polynomials," *Communications on Pure and Applied Mathematics*, vol. 2, no. 4, pp. 325–330, 1949.

[45] B. Holmquist, "The d-variate vector hermite polynomial of order k," *Linear algebra and its applications*, vol. 237, pp. 155–190, 1996.

[46] Z. Ma, M. Xiao, Y. Xiao, Z. Pang, H. V. Poor, and B. Vucetic, "High-reliability and low-latency wireless communication for internet of things: challenges, fundamentals, and enabling technologies," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7946–7970, 2019.

[47] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[48] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland et al., "Protein structure prediction using multiple deep neural networks in the 13th critical assessment of protein structure prediction (casp13)," *Proteins: Structure, Function, and Bioinformatics*, vol. 87, no. 12, pp. 1141–1148, 2019.

[49] S.-M. Udrescu and M. Tegmark, "Ai feynman: A physics-inspired method for symbolic regression," *Science Advances*, vol. 6, no. 16, p. eaay2631, 2020.

[50] M. Plotkin, "Binary codes with specified minimum distance," *IRE Transactions on Information Theory*, vol. 6, no. 4, pp. 445–450, 1960.

[51] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron, "Testing reed-muller codes," *IEEE Transactions on Information Theory*, vol. 51, no. 11, pp. 4032–4039, 2005.

[52] S. Kudekar, S. Kumar, M. Mondelli, H. D. Pfister, E. Şaşoğlu, and R. L. Urbanke, "Reed–muller codes achieve capacity on erasure channels," *IEEE Transactions on information theory*, vol. 63, no. 7, pp. 4298–4316, 2017.

[53] M. Ye and E. Abbe, "Recursive projection-aggregation decoding of reed-muller codes," *IEEE Transactions on Information Theory*, vol. 66, no. 8, pp. 4948–4965, 2020.

[54] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[55] I. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 38–49, 1954.

[56] E. Abbe, A. Shpilka, and M. Ye, "Reed-muller codes: Theory and algorithms," 2020.

[57] I. Dumer, "Recursive decoding and its performance for low-rate reed-muller codes," *IEEE Transactions on Information Theory*, vol. 50, no. 5, pp. 811–823, 2004.

[58] I. Dumer, "Soft-decision decoding of reed-muller codes: a simplified algorithm," *IEEE transactions on information theory*, vol. 52, no. 3, pp. 954–963, 2006.

[59] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[60] H. Kim, S. Oh, and P. Viswanath, "Physical layer communication via deep learning," *IEEE Journal on Selected Areas in Information Theory*, 2020.

[61] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

[62] S. Dörner, S. Cammerer, J. Hoydis, and S. Ten Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2017.

[63] I. Sason and S. Shamai, "Performance analysis of linear codes under maximum-likelihood decoding: A tutorial," 2006.

[64] T. Kaufman, S. Lovett, and E. Porat, "Weight distribution and list-decoding size of reed–muller codes," *IEEE transactions on information theory*, vol. 58, no. 5, pp. 2689–2696, 2012.

[65] E. Abbe, A. Shpilka, and A. Wigderson, "Reed–muller codes for random erasures and errors," *IEEE Transactions on Information Theory*, vol. 61, no. 10, pp. 5229–5252, 2015.

[66] O. Sberlo and A. Shpilka, "On the performance of reed-muller codes with respect to random errors and erasures," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms.* SIAM, 2020, pp. 1357–1376.

[67] Y. Polyanskiy, H. V. Poor, and S. Verdú, "Channel coding rate in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2307–2359, 2010.

[68] I. Dumer and K. Shabunov, "Near-optimum decoding for subcodes of reed-muller codes," in *Proceedings. 2001 IEEE International Symposium on Information Theory.* IEEE, 2001, p. 329.

[69] A. Lapidoth, "Nearest neighbor decoding for additive non-gaussian noise channels," *IEEE Transactions on Information Theory*, vol. 42, no. 5, pp. 1520–1529, 1996.

[70] M. Sybis, K. Wesolowski, K. Jayasinghe, V. Venkatasubramanian, and V. Vukadinovic, "Channel coding for ultra-reliable low-latency communication in 5g systems," in *2016 IEEE 84th vehicular technology conference (VTC-Fall).* IEEE, 2016, pp. 1–5.

[71] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, "Learn codes: Inventing low-latency codes via recurrent neural networks," *IEEE Journal on Selected Areas in Information Theory*, 2020.

[72] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," *arXiv preprint arXiv:1811.04918*, 2018.

[73] C. Stein, "A bound for the error in the normal approximation to the distribution of a sum of dependent random variables," in *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 2. University of California Press, 1972, pp. 583–602.

[74] I.-C. Yeh, "Modeling of strength of high performance concrete using artificial neural networks," *Cement and Concrete Research*, vol. 28, no. 12, pp. 1797–1808, 1998. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

[75] Y.-C. Liu and I.-C. Yeh, "Using mixture design and neural networks to build stock selection decision support systems," *Neural Computing and Applications*, vol. 28, no. 3, pp. 521–535, 2017. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Stock+portfolio+performance

[76] T. Brooks, D. Pope, and A. Marcolini., "Airfoil self-noise and prediction," NASA, Tech. Rep., 1989. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise

[77] M. Ledoux and M. Talagrand, *Probability in Banach Spaces: isoperimetry and processes.* Berlin: Springer, May 1991.

[78] A. W. Vaart and J. A. Wellner, *Weak convergence and empirical processes: with applications to statistics.* Springer, 1996.

[79] I. Tal and A. Vardy, "How to construct polar codes," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6562–6582, 2013.

[80] M. V. Jamali, X. Liu, A. V. Makkuva, H. Mahdavifar, S. Oh, and P. Viswanath, "Reed-Muller subcodes: Machine learning-aided design of efficient soft recursive decoding," *arXiv preprint arXiv:2102.01671*, 2021.

Appendix A

Appendix for Chapter 2

Organization. The appendix is organized as follows:

- Appendix A.1 and Appendix A.1.1 contain the requisite material for method of moments and the convergence analysis of EM respectively.
- Appendix A.2 details the class of non-linearities for which our results hold.
- Appendix A.3 contains all the proofs of Section 2.4. Two technical lemmas needed to prove Theorem 2 are relegated to Appendix A.7 and Appendix A.8.
- Appendix A.11 provides convergence guarantees for Gradient EM.
- Appendix A.12 contains additional experiments for the comparison of joint-EM and our algorithm for the synthetic data.

A.1 Toolbox for method of moments

In this section, we introduce the key techniques that are useful in parameter estimation of mixture models via the method of moments.

Stein's identity (Stein's lemma) is a well-known result in probability and statistics and is widely used in estimation and inference taks. A refined version of the Stein's lemma [73] for higher-order moments is the key to parameter estimation in mixture of generalized linear models. We utilize this machinery in proving Theorem 1. We first recall the Stein's lemma.

Lemma 1 (Stein's lemma [73]). *Let $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$ and $g : \mathbb{R}^d \to \mathbb{R}$ be a function such that both $\mathbb{E}[\nabla_{\boldsymbol{x}} g(\boldsymbol{x})]$ and $\mathbb{E}[g(\boldsymbol{x}) \cdot \boldsymbol{x}]$ exist and are finite. Then*

$$\mathbb{E}[g(\boldsymbol{x}) \cdot \boldsymbol{x}] = \mathbb{E}[\nabla_{\boldsymbol{x}} g(\boldsymbol{x})].$$

The following lemma, which can be viewed as an extension of Stein's lemma for higher-order moments, is the central technique behind parameter estimation in M-GLMs.

Lemma 2 ([23]). *Let $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$ and $\mathcal{S}_3(\boldsymbol{x})$ be as defined in Equation 2.6 and let $\mathcal{S}_2(\boldsymbol{x}) \triangleq \boldsymbol{x} \otimes \boldsymbol{x} - I_d$. Then for any $g : \mathbb{R}^d \to \mathbb{R}$ satisfying some regularity conditions, we have*

$$\mathbb{E}[g(\boldsymbol{x}) \cdot \mathcal{S}_2(\boldsymbol{x})] = \mathbb{E}[\nabla^{(2)}_{\boldsymbol{x}} g(\boldsymbol{x})], \quad \mathbb{E}[g(\boldsymbol{x}) \cdot \mathcal{S}_3(\boldsymbol{x})] = \mathbb{E}[\nabla^{(3)}_{\boldsymbol{x}} g(\boldsymbol{x})].$$

A.1.1 Toolbox for EM convergence analysis

Recall that the domain of our gating parameters is $\Omega = \{\boldsymbol{w} : \|\boldsymbol{w}\| \leq 1\}$. Then the population EM for the mixture of experts consists of the following two steps:

- **E-step:** Using the current estimate $\boldsymbol{w}_t$ to compute the function $Q(\cdot|\boldsymbol{w}_t)$.
- **M-step:** $\boldsymbol{w}_{t+1} = \text{argmax}_{\|\boldsymbol{w}\| \leq 1} Q(\boldsymbol{w}|\boldsymbol{w}_t)$.

Thus the EM can be viewed as a deterministic procedure which maps $\boldsymbol{w}_t \mapsto M(\boldsymbol{w}_t)$ where

$$M(\boldsymbol{w}) = \text{argmax}_{\boldsymbol{w}' \in \Omega} Q(\boldsymbol{w}'|\boldsymbol{w}).$$

Our convergence analysis relies on tools from [27] where they provided local convergence results on both the EM and gradient EM algorithms. In particular, they showed that if we initialize EM in a sufficiently small neighborhood around the true parameters, the EM iterates converge geometrically to the true parameters under some strong-concavity and gradient stability conditions. We now formally state the assumptions in [27] under which the convergence guarantees hold. We will show in the next section that these conditions hold *globally* in our setting.

Assumption 1 (Convexity of the domain)**.** Ω is convex.

Assumption 2 (Strong-concavity)**.** $Q(\cdot|\boldsymbol{w}^*)$ is a λ-strongly concave function over a r-neighborhood of $\boldsymbol{w}^*$, i.e. $\mathcal{B}(\boldsymbol{w}^*, r) \triangleq \{\boldsymbol{w} \in \Omega : \|\boldsymbol{w} - \boldsymbol{w}^*\| \leq r\}$.

Remark 2. An important point to note is that the true parameter $\boldsymbol{w}^*$ is a fixed point for the EM algorithm, i.e. $M(\boldsymbol{w}^*) = \boldsymbol{w}^*$. This is also known as *self-consistency* of the EM algorithm. Hence it is reasonable to expect that in a sufficiently small neighborhood around $\boldsymbol{w}^*$ there exists a unique maximizer for $Q(\cdot|\boldsymbol{w}^*)$.

Assumption 3 (First-order stability condition)**.** Assume that

$$\|\nabla Q(M(\boldsymbol{w})|\boldsymbol{w}^*) - \nabla Q(M(\boldsymbol{w})|\boldsymbol{w})\| \leq \gamma \|\boldsymbol{w} - \boldsymbol{w}^*\|, \quad \forall \boldsymbol{w} \in \mathcal{B}(\boldsymbol{w}^*, r).$$

Remark 3. Intuitively, the gradient stability condition enforces the gradient maps $\nabla Q(\cdot|\boldsymbol{w})$ and $\nabla Q(\cdot|\boldsymbol{w}^*)$ to be close whenever $\boldsymbol{w}$ lies in a neighborhood of $\boldsymbol{w}^*$. This will ensure that the mapped output $M(\boldsymbol{w})$ stays closer to $\boldsymbol{w}^*$.

Theorem 7 (Theorem 1, [27])**.** *If the above assumptions are met for some radius $r > 0$ and $0 \leq \gamma < \lambda$, then the map $\boldsymbol{w} \mapsto M(\boldsymbol{w})$ is contractive over $\mathcal{B}(\boldsymbol{w}^*, r)$, i.e.*

$$\|M(\boldsymbol{w}) - \boldsymbol{w}^*\| \leq \left(\frac{\gamma}{\lambda}\right) \|\boldsymbol{w} - \boldsymbol{w}^*\|, \quad \forall \boldsymbol{w} \in \mathcal{B}(\boldsymbol{w}^*, r),$$

and consequently, the EM iterates $\{\boldsymbol{w}_t\}_{t\geq 0}$ converge geometrically to $\boldsymbol{w}^*$, i.e.

$$\|\boldsymbol{w}_t - \boldsymbol{w}^*\| \leq \left(\frac{\gamma}{\lambda}\right)^t \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|,$$

whenever the initialization $\boldsymbol{w}_0 \in \mathcal{B}(\boldsymbol{w}^*, r)$.

A.2 Class of non-linearities

In this section, we characterize the class of non-linearities for which our theoretical results for the recovery of regressors hold. Let $Z \sim \mathcal{N}(0, 1)$ and $Y|Z \sim \mathcal{N}(g(Z), \sigma^2)$, where $g : \mathbb{R} \to \mathbb{R}$. For $(\alpha, \beta, \gamma) \in \mathbb{R}^3$, define

$$\begin{aligned}
&\mathcal{P}_3(y) \triangleq Y^3 + \alpha Y^2 + \beta Y, \\
&\mathcal{S}_3(Z) = \mathbb{E}[\mathcal{P}_3(y)|Z] = g(Z)^3 + \alpha g(Z)^2 + g(Z)(\beta + 3\sigma^2) + \alpha\sigma^2,
\end{aligned}$$

and

$$\mathcal{S}_2(Y) \triangleq Y^2 + \gamma Y, \quad \mathcal{S}_2(Z) = \mathbb{E}[\mathcal{S}_2(Y)|Z] = g(Z)^2 + \gamma g(Z) + \sigma^2.$$

Condition 1. $\mathbb{E}[\mathcal{S}_3'(Z)] = \mathbb{E}[\mathcal{S}_3''(Z)] = 0$ and $\mathbb{E}[\mathcal{S}_3'''(Z)] \neq 0$.

Condition 2. $\mathbb{E}[\mathcal{S}_2'(Z)] = 0$ and $\mathbb{E}[\mathcal{S}_2''(Z)] \neq 0$.

We are now ready to define the (α, β, γ)-valid class of non-linearities.

Definition 1. *We say that the non-linearity g is (α, β, γ)-valid if there exists $(\alpha, \beta, \gamma) \in \mathbb{R}^3$ such that both Condition 3 and Condition 4 are satisfied.*

We have that

$$\begin{aligned}
\mathcal{S}_3'(Z) &= 3g(Z)^2 g'(Z) + 2\alpha g(Z)g'(Z) + g'(Z)(\beta + 3\sigma^2) \\
&= 2\alpha g(Z)g'(Z) + \beta g'(Z) + 3g(Z)^2 g'(Z) + 3g'(Z)\sigma^2, \\
\mathcal{S}_3''(Z) &= 2\alpha\left(g'(Z)^2 + g(Z)g''(Z)\right) + \beta g''(Z) + 3g''(Z)(g(Z)^2 + \sigma^2) + 6g(Z)g'(Z)^2.
\end{aligned}$$

Thus $\mathbb{E}[\mathcal{S}_3'(Z)] = \mathbb{E}[\mathcal{S}_3''(Z)] = 0$ implies that

$$\begin{bmatrix} 2\mathbb{E}(g(Z)g'(Z)) & \mathbb{E}(g'(Z)) \\ 2\mathbb{E}\left(g'(Z)^2 + g(Z)g''(Z)\right) & \mathbb{E}(g''(Z)) \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -3\mathbb{E}(g(Z)^2 g'(Z) + g'(Z)\sigma^2) \\ -3\mathbb{E}(g''(Z)(g(Z)^2 + \sigma^2) + 2g(Z)g'(Z)^2) \end{bmatrix}$$

To ensure Condition 3, we need the pair (α, β) obtained by solving the above linear equation to satisfy $\mathbb{E}[\mathcal{S}_3'''(Z)] \neq 0$. Similarly, $\mathbb{E}[\mathcal{S}_2'(Z)] = 0$ implies that

$$\gamma = \frac{-2\mathbb{E}[g(Z)g'(Z)]}{\mathbb{E}[g'(Z)]}.$$

Thus Condition 4 stipulates that $\mathbb{E}[\mathcal{S}_2''(Z)] \neq 0$ with this choice of γ. It turns out that these conditions hold for a wide class of non-linearities and in particular, when g is either the identity function, or the sigmoid function, or the ReLU. For these three choices of popular non-linearities, the values of the tuple (α, β, γ) are provided below (which are obtained by solving the linear equations mentioned above).

Example 1. If g is the identity mapping, then $\mathcal{P}_3(y) = y^3 - 3y(1 + \sigma^2)$ and $\mathcal{S}_2(y) = y^2$.

Example 2. If g is the sigmoid function, i.e. $g(z) = \frac{1}{1+e^{-z}}$, then α and β can be obtained by solving the following linear equation:

$$\begin{bmatrix} 0.2066 & 0.2066 \\ 0.0624 & -0.0001 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -0.1755 - 0.6199\sigma^2 \\ -0.0936 \end{bmatrix}$$

The second-order transformation is given by $\mathcal{S}_2(y) = y^2 - y$ (since $\gamma = -1$ when g is sigmoid).

Example 3. If g is the ReLU function, i.e. $g(z) = \max\{0, z\}$, then $\alpha = -3\sqrt{\frac{2}{\pi}}, \beta = 3\left(\frac{4}{\pi} - \sigma^2 - 1\right)$ and $\gamma = -2\sqrt{\frac{2}{\pi}}$.

A.3 Proofs of Section 2.4

In this section, for the simplicity of the notation we denote the true parameters as $\boldsymbol{w}_i$'s and $\boldsymbol{a}_i$'s dropping the $*$ sign.

A.4 Proof of Theorem 1 for $k = 2$

Proof. Suppose that g is the linear activation function. For $k = 2$, Equation 2.1 implies that

$$P_{y|\boldsymbol{x}} = f(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \mathcal{N}(y|\boldsymbol{a}_1^\top \boldsymbol{x}, \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x})) \cdot \mathcal{N}(y|\boldsymbol{a}_2^\top \boldsymbol{x}, \sigma^2), \quad \boldsymbol{x} \sim \mathcal{N}(0, I_d), \tag{A.1}$$

where $f(\cdot)$ is the sigmoid function. Using the fact $\mathbb{E}[Z^3] = \mu^3 + 3\mu\sigma^2$ for any Gaussian random variable $Z \sim \mathcal{N}(\mu, \sigma^2)$, we get

$$\mathbb{E}[y^3|\boldsymbol{x}] = f(\boldsymbol{w}^\top \boldsymbol{x})((\boldsymbol{a}_1^\top \boldsymbol{x})^3 + 3(\boldsymbol{a}_1^\top \boldsymbol{x})\sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x}))((\boldsymbol{a}_1^\top \boldsymbol{x})^3 + 3(\boldsymbol{a}_1^\top \boldsymbol{x})\sigma^2).$$

Moreover,

$$\mathbb{E}[y|\boldsymbol{x}] = f(\boldsymbol{w}^\top \boldsymbol{x})(\boldsymbol{a}_1^\top \boldsymbol{x}) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x}))(\boldsymbol{a}_2^\top \boldsymbol{x}).$$

Thus,

$$\mathbb{E}[y^3 - 3y(1+\sigma^2)|\boldsymbol{x}] = f(\boldsymbol{w}^\top \boldsymbol{x})((\boldsymbol{a}_1^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_1^\top \boldsymbol{x})) \\ + (1 - f(\boldsymbol{w}^\top \boldsymbol{x}))((\boldsymbol{a}_1^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_1^\top \boldsymbol{x})).$$

If we define $\mathcal{P}_3(y) \triangleq y^3 - 3y(1+\sigma^2)$, in view of Lemma 2 we get that

$$\begin{aligned} \mathcal{T}_3 = \mathbb{E}[\mathcal{P}_3(y)\cdot \mathcal{S}_3(\boldsymbol{x})] &= \mathbb{E}[(y^3 - 3y(1+\sigma^2))\cdot \mathcal{S}_3(\boldsymbol{x})] \\ &= \mathbb{E}\left[\left(f(\boldsymbol{w}^\top \boldsymbol{x})((\boldsymbol{a}_1^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_1^\top \boldsymbol{x}))\right)\cdot \mathcal{S}_3(\boldsymbol{x})\right] \qquad (A.2) \\ &\quad + \mathbb{E}\left[\left(1 - f(\boldsymbol{w}^\top \boldsymbol{x})((\boldsymbol{a}_2^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_2^\top \boldsymbol{x}))\right)\cdot \mathcal{S}_3(\boldsymbol{x})\right] \\ &= \mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)}\left(f(\boldsymbol{w}^\top \boldsymbol{x})((\boldsymbol{a}_1^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_1^\top \boldsymbol{x}))\right)\right] \qquad (A.3) \\ &\quad + \mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)}\left(1 - f(\boldsymbol{w}^\top \boldsymbol{x})((\boldsymbol{a}_2^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_2^\top \boldsymbol{x}))\right)\right]. \\ &\qquad (A.4) \end{aligned}$$

Using the chain rule for multi-derivatives, the first term simplifies to

$$\begin{aligned} \mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)}\left(f(\boldsymbol{w}^\top \boldsymbol{x})((\boldsymbol{a}_1^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_1^\top \boldsymbol{x}))\right)\right] &= \mathbb{E}[f'''((\boldsymbol{a}_1^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_1^\top \boldsymbol{x}))]\cdot \boldsymbol{w}\otimes\boldsymbol{w}\otimes\boldsymbol{w} \\ &\quad + \mathbb{E}[f''(3(\boldsymbol{a}_1^\top \boldsymbol{x})^2 - 3)]\cdot \\ &\quad (\boldsymbol{w}\otimes\boldsymbol{w}\otimes\boldsymbol{a}_1 + \boldsymbol{w}\otimes\boldsymbol{a}_1\otimes\boldsymbol{w} \\ &\quad + \boldsymbol{a}_1\otimes\boldsymbol{w}\otimes\boldsymbol{w}) + \\ &\quad \mathbb{E}[f'(6(\boldsymbol{a}_1^\top \boldsymbol{x}))]\cdot(\boldsymbol{a}_1\otimes\boldsymbol{a}_1\otimes\boldsymbol{w} + \boldsymbol{a}_1\otimes\boldsymbol{w}\otimes\boldsymbol{a}_1 + \boldsymbol{w}\otimes\boldsymbol{a}_1\otimes\boldsymbol{a}_1) \\ &\quad + 6\mathbb{E}[f]\cdot \boldsymbol{a}_1\otimes\boldsymbol{a}_1\otimes\boldsymbol{a}_1. \\ &\qquad (A.5) \end{aligned}$$

Since $f(z) = \frac{1}{1+e^{-z}}$, $f'(\cdot), f'''(\cdot)$ are even functions whereas $f''(\cdot)$ is an odd function. Furthermore, both $\boldsymbol{x} \mapsto (\boldsymbol{a}_1^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_1^\top \boldsymbol{x})$ and $\boldsymbol{x} \mapsto \boldsymbol{a}_1^\top \boldsymbol{x}$ are odd functions whereas $\boldsymbol{x} \mapsto 3(\boldsymbol{a}_1^\top \boldsymbol{x})^2 - 3$ is an even function. Since $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$, $-\boldsymbol{x} \overset{(d)}{=} \boldsymbol{x}$. Thus all the expectation terms in Equation A.5 equal zero except for the last term since $\mathbb{E}[f(\boldsymbol{w}^\top \boldsymbol{x})] = \frac{1}{2} > 0$. We have,

$$\mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)}\left(f(\boldsymbol{w}^\top \boldsymbol{x})((\boldsymbol{a}_1^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_1^\top \boldsymbol{x}))\right)\right] = 3\cdot \boldsymbol{a}_1\otimes\boldsymbol{a}_1\otimes\boldsymbol{a}_1.$$

Similarly,

$$\mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)}\left(1 - f(\boldsymbol{w}^\top \boldsymbol{x})((\boldsymbol{a}_2^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_2^\top \boldsymbol{x}))\right)\right] = 3\cdot \boldsymbol{a}_2\otimes\boldsymbol{a}_2\otimes\boldsymbol{a}_2.$$

Together, we have that

$$\mathcal{T}_3 = 3 \cdot \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 + 3 \cdot \boldsymbol{a}_2 \otimes \boldsymbol{a}_2 \otimes \boldsymbol{a}_2.$$

Now consider an arbitrary link function g belonging to the class of non-linearities described in Appendix A.2. Then

$$P_{y|\boldsymbol{x}} = f(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x})) \cdot \mathcal{N}(y|g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2), \quad \boldsymbol{x} \sim \mathcal{N}(0, I_d),$$

implies that

$$\mathbb{E}[y^3|\boldsymbol{x}] = f(\boldsymbol{w}^\top \boldsymbol{x})(g(\boldsymbol{a}_1^\top \boldsymbol{x})^3 + 3g(\boldsymbol{a}_1^\top \boldsymbol{x})\sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x}))(g(\boldsymbol{a}_2^\top \boldsymbol{x})^3 + 3g(\boldsymbol{a}_2^\top \boldsymbol{x})\sigma^2),$$

and

$$\begin{aligned}
\mathbb{E}[y^2|\boldsymbol{x}] &= f(\boldsymbol{w}^\top \boldsymbol{x})(g(\boldsymbol{a}_1^\top \boldsymbol{x})^2 + \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x}))(g(\boldsymbol{a}_2^\top \boldsymbol{x})^2 + \sigma^2), \\
\mathbb{E}[y|\boldsymbol{x}] &= f(\boldsymbol{w}^\top \boldsymbol{x}) g(\boldsymbol{a}_1^\top \boldsymbol{x}) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x})) g(\boldsymbol{a}_2^\top \boldsymbol{x}).
\end{aligned}$$

If we define $\mathcal{P}_3(y) \triangleq y^3 + \alpha y^2 + \beta y$, we have that

$$\begin{aligned}
\mathcal{T}_3 &= \mathbb{E}[\mathcal{P}_3(y) \cdot \mathcal{S}_3(\boldsymbol{x})] \\
&= \mathbb{E}[\mathbb{E}[y^3 + \alpha y^2 + \beta y|\boldsymbol{x}] \cdot \mathcal{S}_3(\boldsymbol{x})] \\
&= \mathbb{E}\left[f(\boldsymbol{w}^\top \boldsymbol{x})\left(g(\boldsymbol{a}_1^\top \boldsymbol{x})^3 + \alpha g(\boldsymbol{a}_1^\top \boldsymbol{x})^2 + g(\boldsymbol{a}_1^\top \boldsymbol{x})(\beta + 3\sigma^2)\right) \cdot \mathcal{S}_3(\boldsymbol{x})\right] + \\
&\quad \mathbb{E}\left[(1 - f(\boldsymbol{w}^\top \boldsymbol{x}))\left(g(\boldsymbol{a}_2^\top \boldsymbol{x})^3 + \alpha g(\boldsymbol{a}_2^\top \boldsymbol{x})^2 + g(\boldsymbol{a}_2^\top \boldsymbol{x})(\beta + 3\sigma^2)\right) \cdot \mathcal{S}_3(\boldsymbol{x})\right] \\
&= \mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)}\left(f(\boldsymbol{w}^\top \boldsymbol{x})\left(g(\boldsymbol{a}_1^\top \boldsymbol{x})^3 + \alpha g(\boldsymbol{a}_1^\top \boldsymbol{x})^2 + g(\boldsymbol{a}_1^\top \boldsymbol{x})(\beta + 3\sigma^2)\right)\right)\right] + \\
&\quad \mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)}\left(f(\boldsymbol{w}^\top \boldsymbol{x})\left(g(\boldsymbol{a}_2^\top \boldsymbol{x})^3 + \alpha g(\boldsymbol{a}_2^\top \boldsymbol{x})^2 + g(\boldsymbol{a}_2^\top \boldsymbol{x})(\beta + 3\sigma^2)\right)\right)\right] \\
&\overset{(a)}{=} \mathbb{E}[f]\mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)}\left(g(\boldsymbol{a}_1^\top \boldsymbol{x})^3 + \alpha g(\boldsymbol{a}_1^\top \boldsymbol{x})^2 + g(\boldsymbol{a}_1^\top \boldsymbol{x})(\beta + 3\sigma^2)\right)\right] \cdot \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 + \\
&\quad \mathbb{E}[1 - f]\mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)}\left(g(\boldsymbol{a}_2^\top \boldsymbol{x})^3 + \alpha g(\boldsymbol{a}_2^\top \boldsymbol{x})^2 + g(\boldsymbol{a}_2^\top \boldsymbol{x})(\beta + 3\sigma^2)\right)\right] \cdot \boldsymbol{a}_2 \otimes \boldsymbol{a}_2 \otimes \boldsymbol{a}_2 \\
&= c_{g,\sigma}\left(\mathbb{E}[f] \cdot \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 \otimes \boldsymbol{a}_1 + \mathbb{E}[1 - f] \cdot \boldsymbol{a}_2 \otimes \boldsymbol{a}_2 \otimes \boldsymbol{a}_2\right),
\end{aligned}$$

where (a) follows from the choice of α and β and the fact that $\boldsymbol{w} \perp \{\boldsymbol{a}_1, \boldsymbol{a}_2\}$, and $c_{g,\sigma} \triangleq \mathbb{E}\left[(g(Z)^3 + \alpha g(Z)^2 + g(Z)(\beta + 3\sigma^2))'''\right]$ where $Z \sim \mathcal{N}(0, 1)$. The proof for $\mathcal{T}_2$ is similar.

□

A.5 Proof of Theorem 1 for general k

Proof. The proof for general k closely follows that of $k = 2$, described in Appendix A.4. For the general k, we first prove the theorem when g is the identity function, i.e.

$$P_{y|\boldsymbol{x}} = \sum_{i\in[k]} P_{i|\boldsymbol{x}} P_{y|\boldsymbol{x},i} = \sum_{i\in[k]} \frac{e^{\boldsymbol{w}_i^\top \boldsymbol{x}}}{\sum_{i\in[k]} e^{\boldsymbol{w}_i^\top \boldsymbol{x}}} \cdot \mathcal{N}(y|\boldsymbol{a}_i^\top \boldsymbol{x}, \sigma^2), \quad \boldsymbol{x} \sim \mathcal{N}(0, I_d).$$

Denoting $P_{i|\boldsymbol{x}}$ by $p_i(\boldsymbol{x})$, we have that

$$\begin{aligned}\mathbb{E}[y^3|\boldsymbol{x}] &= \sum_{i\in[k]} p_i(\boldsymbol{x}) \left((\boldsymbol{a}_i^\top \boldsymbol{x})^3 + 3(\boldsymbol{a}_i^\top \boldsymbol{x})\sigma^2\right), \\ \mathbb{E}[y|\boldsymbol{x}] &= \sum_{i\in[k]} p_i(\boldsymbol{x})(\boldsymbol{a}_i^\top \boldsymbol{x}).\end{aligned}$$

Hence

$$\mathbb{E}[y^3 - 3y(1+\sigma^2)|\boldsymbol{x}] = \sum_{i\in[k]} p_i(\boldsymbol{x}) \left((\boldsymbol{a}_i^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_i^\top \boldsymbol{x})\right)$$

If we let $\mathcal{P}_3(y) \triangleq y^3 - 3y(1+\sigma^2)$, we get

$$\mathbb{E}[\mathcal{P}_3(y) \cdot \mathcal{S}_3(\boldsymbol{x})] = \sum_{i\in[k]} \mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)} \left(p_i(\boldsymbol{x}) \left((\boldsymbol{a}_i^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_i^\top \boldsymbol{x})\right)\right)\right]$$

Since $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$ and $\boldsymbol{a}_i \perp \text{span}\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{k-1}\}$, we have that $\boldsymbol{a}_i^\top \boldsymbol{x} \perp (\boldsymbol{w}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{w}_{k-1}^\top \boldsymbol{x})$. Moreover, $\mathbb{E}[(\boldsymbol{a}_i^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_i^\top \boldsymbol{x})] = \mathbb{E}[(\boldsymbol{a}_i^\top \boldsymbol{x})^2 - 1] = \mathbb{E}[\boldsymbol{a}_i^\top \boldsymbol{x}] = 0$ for each i. Using the chain-rule for multi-derivatives, the above equation thus simplifies to

$$\begin{aligned}\mathbb{E}[\mathcal{P}_3(y) \cdot \mathcal{S}_3(\boldsymbol{x})] &= \sum_{i\in[k]} \mathbb{E}[p_i(\boldsymbol{x})] \cdot \mathbb{E}\left[\nabla_{\boldsymbol{x}}^{(3)} \left((\boldsymbol{a}_i^\top \boldsymbol{x})^3 - 3(\boldsymbol{a}_i^\top \boldsymbol{x})\right)\right] \\ &= \sum_{i\in[k]} 6\mathbb{E}[p_i(\boldsymbol{x})] \cdot \boldsymbol{a}_i \otimes \boldsymbol{a}_i \times \boldsymbol{a}_i.\end{aligned}$$

For a generic $g : \mathbb{R} \to \mathbb{R}$ which is $(\alpha, \beta, \gamma)-$valid, let $\mathcal{P}_3(y) = y^3 + \alpha y^2 + \beta y$. Then it is easy to see that the same proof goes through except for a change in the coefficients of rank-1 terms, i.e.

$$\mathbb{E}[\mathcal{P}_3(y) \cdot \mathcal{S}_3(\boldsymbol{x})] = \sum_{i \in [k]} \alpha_i \mathbb{E}[p_i(\boldsymbol{x})] \cdot \boldsymbol{a}_i \otimes \boldsymbol{a}_i \otimes \boldsymbol{a}_i,$$

where $\alpha_i \triangleq \mathbb{E}\left[(g(Z)^3 + \alpha g(Z)^2 + g(Z)(\beta + 3\sigma^2))'''\right]$ where $Z \sim \mathcal{N}(0,1)$ and $'''$ denotes the third-derivative with respect to Z. Note that Condition 4 together with the fact that $\mathbb{E}[p_i(\boldsymbol{x})] > 0$ ensures that $\alpha_i \neq 0$ and thus the coefficients of the rank-1 terms are non-zero. The proof for $\mathcal{T}_2$ is similar. $\square$

A.6 Proof of Theorem 2

The following two lemmas are central to the proof of Theorem 2. Let $\boldsymbol{A}^\top = [\boldsymbol{a}_1 | \ldots | \boldsymbol{a}_k] \in \mathbb{R}^{d \times k}$ denote the matrix of regressor parameters whereas $\boldsymbol{W}^\top = [\boldsymbol{w}_1 | \ldots | \boldsymbol{w}_{k-1}] \in \mathbb{R}^{d \times (k-1)}$ denote the matrix of gating parameters. With a slight change of notation, when $\boldsymbol{A} = \boldsymbol{A}^*$, we denote the EM operator $M(\boldsymbol{W})$ as either $M(\boldsymbol{W}, \boldsymbol{A}^*)$ or $M(\boldsymbol{w})$, introduced in Section 2.4. For the general case, we simply denote it by $M(\boldsymbol{W}, \boldsymbol{A})$. In the following lemmas, we use the norm $\|\boldsymbol{A}\| = \max_{i \in [k]} \|\boldsymbol{A}_i^\top\|_2$ where $\boldsymbol{A} \in \mathbb{R}^{k \times d}$ is a matrix of regressors, similarly for any matrix of classifiers $\boldsymbol{W} \in \mathbb{R}^{(k-1) \times d}$.

Lemma 3 (Contraction of the EM operator)**.** *Under the assumptions of Theorem 2, we have that*

$$\|M(\boldsymbol{W}, \boldsymbol{A}^*) - \boldsymbol{W}^*\| \leq \kappa_\sigma \|\boldsymbol{W} - \boldsymbol{W}^*\|.$$

Moreover, $\boldsymbol{W} = \boldsymbol{W}^*$ is a fixed point for $M(\boldsymbol{W}, \boldsymbol{A}^*)$.

Lemma 4 (Robustness of the EM operator)**.** *Let the matrix of regressors $\boldsymbol{A}$ be such that $\max_{i \in [k]} \|\boldsymbol{A}_i^\top - (\boldsymbol{A}_i^*)^\top\|_2 = \sigma^2 \varepsilon_1$. Then for any $\boldsymbol{W} \in \Omega$, we have that*

$$\|M(\boldsymbol{W}, \boldsymbol{A}) - M(\boldsymbol{W}, \boldsymbol{A}^*)\| \leq \kappa \varepsilon_1,$$

where κ is a constant depending on g, k and σ. In particular, $\kappa \leq (k-1)\frac{\sqrt{6(2+\sigma^2)}}{2}$ for $g =$linear, sigmoid and ReLU.

We are now ready to prove Theorem 2.

Proof. We first note that the EM iterates $\{\boldsymbol{W}_t\}_{t\geq 1}$ evolve according to

$$\boldsymbol{W}_t = M(\boldsymbol{W}_{t-1}, \boldsymbol{A}), \quad t \geq 1$$

Thus

$$\begin{aligned}\|\boldsymbol{W}_t - \boldsymbol{W}^*\| = \|M(\boldsymbol{W}_{t-1}, \boldsymbol{A}) - \boldsymbol{W}^*\| &= \|M(\boldsymbol{W}_{t-1}, \boldsymbol{A}) - M(\boldsymbol{W}^*, \boldsymbol{A}^*)\| \\ &\leq \|M(\boldsymbol{W}_{t-1}, \boldsymbol{A}) - M(\boldsymbol{W}_{t-1}, \boldsymbol{A}^*)\| \\ &\quad + \|M(\boldsymbol{W}_{t-1}, \boldsymbol{A}^*) - \boldsymbol{W}^*\| \\ &\leq k\varepsilon_1 + \kappa_\sigma \|\boldsymbol{W}_{t-1} - \boldsymbol{W}^*\|,\end{aligned}$$

where the last inequality follows from Lemma 7 and Lemma 8. Recursively using the above inequality, we obtain that

$$\begin{aligned}\|\boldsymbol{W}_t - \boldsymbol{W}^*\| &\leq (\kappa_\sigma)^t \|\boldsymbol{W}_0 - \boldsymbol{W}^*\| + \kappa\varepsilon_1(1 + \kappa_\sigma + \ldots + \kappa_\sigma^{t-1}) \\ &\leq (\kappa_\sigma)^t \|\boldsymbol{W}_0 - \boldsymbol{W}^*\| + \frac{\kappa\varepsilon_1}{1-\kappa_\sigma}.\end{aligned}$$

□

A.7 Proof of Lemma 8

We need the following lemma which establishes the stability of the minimizers for strongly convex functions under Lipschitz perturbations.

Lemma 5. *Suppose $\Omega \subseteq \mathbb{R}^d$ is a closed convex subset, $f : \Omega \to \mathbb{R}$ is a λ-strongly convex function for some $\lambda > 0$ and B is an L-Lipschitz continuous function on Ω. Let $\boldsymbol{w}_f = \operatorname{argmin}_{\boldsymbol{w}\in\Omega} f(\boldsymbol{w})$ and $\boldsymbol{w}_{f+B} = \operatorname{argmin}_{\boldsymbol{w}\in\Omega} f(\boldsymbol{w}) + B(\boldsymbol{w})$. Then*

$$\|\boldsymbol{w}_f - \boldsymbol{w}_{f+B}\| \leq \frac{L}{\lambda}.$$

Proof. Let $\boldsymbol{w}' \in \Omega$ be such that $\|\boldsymbol{w}' - \boldsymbol{w}_f\| > \frac{L}{\lambda}$. Let $\boldsymbol{w}_\alpha = \alpha \boldsymbol{w}_f + (1-\alpha)\boldsymbol{w}'$ for $0 < \alpha < 1$. From the fact that $\boldsymbol{w}_f$ is the minimizer of f on Ω and that f is strongly convex, we have that

$$f(\boldsymbol{w}') \geq f(\boldsymbol{w}_f) + \frac{\lambda \|\boldsymbol{w}' - \boldsymbol{w}_f\|^2}{2}.$$

Furthermore, the strong-convexity of f implies that

$$\begin{aligned} f(\boldsymbol{w}_\alpha) &\leq \alpha f(\boldsymbol{w}_f) + (1-\alpha) f(\boldsymbol{w}') - \frac{\alpha(1-\alpha)\lambda}{2}\|\boldsymbol{w}' - \boldsymbol{w}_f\|^2 \\ &= f(\boldsymbol{w}') + \alpha(f(\boldsymbol{w}_f) - f(\boldsymbol{w}')) - \frac{\alpha(1-\alpha)\lambda}{2}\|\boldsymbol{w}' - \boldsymbol{w}_f\|^2 \\ &\leq f(\boldsymbol{w}') - \alpha\frac{\lambda\|\boldsymbol{w}' - \boldsymbol{w}_f\|^2}{2} - \frac{\alpha(1-\alpha)\lambda}{2}\|\boldsymbol{w}' - \boldsymbol{w}_f\|^2 \\ &= f(\boldsymbol{w}') - \lambda\alpha\left(1 - \frac{\alpha}{2}\right)\|\boldsymbol{w}' - \boldsymbol{w}_f\|^2 \end{aligned} \tag{A.6}$$

Since B is L-Lipschitz, we have

$$B(\boldsymbol{w}_\alpha) \leq B(\boldsymbol{w}') + L\alpha\|\boldsymbol{w}' - \boldsymbol{w}_f\|. \tag{A.7}$$

Adding Equation A.6 and Equation A.7, we get

$$\begin{aligned} f(\boldsymbol{w}_\alpha) + B(\boldsymbol{w}_\alpha) &\leq f(\boldsymbol{w}') + B(\boldsymbol{w}') + L\alpha\|\boldsymbol{w}' - \boldsymbol{w}_f\| - \lambda\alpha\left(1 - \frac{\alpha}{2}\right)\|\boldsymbol{w}' - \boldsymbol{w}_f\|^2 \\ &= f(\boldsymbol{w}') + B(\boldsymbol{w}') + \alpha\lambda\|\boldsymbol{w}' - \boldsymbol{w}_f\|\left(\frac{L}{\lambda} - \left(1 - \frac{\alpha}{2}\right)\|\boldsymbol{w}' - \boldsymbol{w}_f\|\right) \end{aligned}$$

By the assumption that $\|\boldsymbol{w}' - \boldsymbol{w}_f\| > \frac{L}{\lambda}$, the term $\frac{L}{\lambda} - \left(1 - \frac{\alpha}{2}\right)\|\boldsymbol{w}' - \boldsymbol{w}_f\|$ will be negative for sufficiently small α. This in turn implies that $f(\boldsymbol{w}_\alpha) + B(\boldsymbol{w}_\alpha) < f(\boldsymbol{w}') + B(\boldsymbol{w}')$ for such α. Consequently $\boldsymbol{w}'$ is not a minimizer of $f + B$ for any $\boldsymbol{w}'$ such that $\|\boldsymbol{w}' - \boldsymbol{w}_f\| > \frac{L}{\lambda}$. The conclusion follows.

□

We are now ready to prove Lemma 8. Fix any $\boldsymbol{W} \in \Omega$ and let $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1^\top \\ \dots \\ \boldsymbol{a}_k^\top \end{bmatrix} \in \mathbb{R}^{k \times d}$ be such that $\max_{i \in [k]} \|\boldsymbol{a}_i - \boldsymbol{a}_i^*\|_2 = \sigma^2 \varepsilon_1$ for some $\varepsilon_1 > 0$. Let

$$\boldsymbol{W}' = M(\boldsymbol{W}, \boldsymbol{A}), \quad (\boldsymbol{W}')^* = M(\boldsymbol{W}, \boldsymbol{A}^*),$$

where,

$$M(\boldsymbol{W},\boldsymbol{A}) = \arg\max_{\boldsymbol{W}'\in\Omega} Q(\boldsymbol{W}'|\boldsymbol{W},\boldsymbol{A}),$$

and,

$$Q(\boldsymbol{W}'|\boldsymbol{W},\boldsymbol{A}) = \mathbb{E}\left[\sum_{i\in[k-1]} p^{(i)}(\boldsymbol{W},\boldsymbol{A})((\boldsymbol{W}'_i)^\top \boldsymbol{x}) - \log\left(1+\sum_{i\in[k-1]} e^{(\boldsymbol{W}'_i)^\top \boldsymbol{x}}\right)\right].$$

Here $p^{(i)}(\boldsymbol{A},\boldsymbol{W}) \triangleq \frac{p_i(\boldsymbol{x})N_i}{\sum_{i\in[k]} p_i(\boldsymbol{x})N_i}$ denotes the posterior probability of choosing the i^{th} expert, where

$$p_i(\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_i^\top \boldsymbol{x}}}{1+\sum_{k\in[k-1]} e^{\boldsymbol{w}_j^\top \boldsymbol{x}}}, \quad N_i \triangleq \mathcal{N}(y|g(\boldsymbol{a}_i^\top \boldsymbol{x}),\sigma^2), \quad N_i^* = \mathcal{N}(y|g((\boldsymbol{a}_i^*)^\top \boldsymbol{x}),\sigma^2).$$

Since both $Q(\cdot|\boldsymbol{W},\boldsymbol{A})$ and $Q(\cdot|\boldsymbol{W},\boldsymbol{A}^*)$ are strongly concave functions over Ω with some strong-concavity parameter λ, Lemma 5 implies that

$$\|M(\boldsymbol{W},\boldsymbol{A}) - M(\boldsymbol{W},\boldsymbol{A}^*)\| \le \frac{L}{\lambda},$$

where L is the Lipschitz-constant for the function $l(\cdot) \triangleq Q(\cdot|\boldsymbol{W},\boldsymbol{A}) - Q(\cdot|\boldsymbol{W},\boldsymbol{A}^*)$. We have that

$$l(\boldsymbol{W}') = \sum_{i\in[k-1]} \mathbb{E}[(p^{(i)}(\boldsymbol{W},\boldsymbol{A}) - p^{(i)}(\boldsymbol{W},\boldsymbol{A}^*)(\boldsymbol{W}'_i)^\top \boldsymbol{x})]$$

Without loss of generality let $i=1$. Since $l(\cdot)$ is linear in $\boldsymbol{W}'$, it suffices to show for each i that

$$\|\mathbb{E}[(p^{(1)}(\boldsymbol{W},\boldsymbol{A}) - p^{(1)}(\boldsymbol{W},\boldsymbol{A}^*)\boldsymbol{x}]\| \le L,$$

We show that $L = \kappa\varepsilon_1$, or equivalently,

$$\|\mathbb{E}[(p^{(1)}(\boldsymbol{W},\boldsymbol{A}) - p^{(1)}(\boldsymbol{W},\boldsymbol{A}^*)\boldsymbol{x}]\| \le \kappa\varepsilon_1,$$

Let

$$\boldsymbol{A}_t = \boldsymbol{A}^* + t\Delta, \quad \Delta = \boldsymbol{A} - \boldsymbol{A}^* \in \mathbb{R}^{k\times d}.$$

By hypothesis, we have that $\|\Delta_i\|_2 \leq \sigma^2 \varepsilon_1$ for all $i \in [k]$. Thus in order to show that

$$\|\mathbb{E}[(p^{(1)}(\boldsymbol{A}, \boldsymbol{W}) - p^{(1)}(\boldsymbol{A}^*, \boldsymbol{W}))\boldsymbol{x}]\|_2 \leq \kappa \varepsilon_1,$$

it suffices to show that

$$\langle \mathbb{E}[(p^{(1)}(\boldsymbol{A}, \boldsymbol{W}) - p^{(1)}(\boldsymbol{A}^*, \boldsymbol{W}))\boldsymbol{x}], \widetilde{\Delta} \rangle \leq \kappa \|\Delta/\sigma^2\|_2 \|\widetilde{\Delta}\|_2, \quad \text{for all } \widetilde{\Delta} \in \mathbb{R}^d.$$

Or equivalently,

$$\mathbb{E}[(p^{(1)}(\boldsymbol{A}, \boldsymbol{W}) - p^{(1)}(\boldsymbol{A}^*, \boldsymbol{W}))\langle \boldsymbol{x}, \widetilde{\Delta} \rangle] \leq \kappa \|\Delta/\sigma^2\|_2 \|\widetilde{\Delta}\|_2.$$

We can rewrite the difference of the posteriors as

$$\begin{aligned} p^{(1)}(\boldsymbol{A}, \boldsymbol{W}) - p^{(1)}(\boldsymbol{A}^*, \boldsymbol{W}) &= \int_0^1 \frac{d}{dt} p^{(1)}(\boldsymbol{A}^* + t\Delta, \boldsymbol{W}) dt && \text{(A.8)} \\ &= \sum_{i \in [k]} \int_0^1 \langle \nabla_{\boldsymbol{a}_i} p^{(1)}(\boldsymbol{A}_t, \boldsymbol{W}), \Delta_i \rangle dt. && \text{(A.9)} \end{aligned}$$

Since $N_i = \mathcal{N}(y | g(\boldsymbol{a}_i^\top \boldsymbol{x}), \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y - g(\boldsymbol{a}_1^\top \boldsymbol{x}))^2 / 2\sigma^2}$, we have that

$$\nabla_{\boldsymbol{a}_i} N_i = N_i \left(\frac{y - g(\boldsymbol{a}_i^\top \boldsymbol{x})}{\sigma^2} \right) g'(\boldsymbol{a}_i^\top \boldsymbol{x}).$$

Thus,

$$\begin{aligned} \nabla_{\boldsymbol{a}_i} p^{(1)}(\boldsymbol{A}_t, \boldsymbol{W}) &= \nabla_{\boldsymbol{a}_i} \left(\frac{p_1(\boldsymbol{x}) N_1}{\sum_{i \in [k]} p_i(\boldsymbol{x}) N_i} \right) \\ &= \begin{cases} \frac{(\sum_{i \neq 1} p_i(\boldsymbol{x}) N_i) p_1(\boldsymbol{x}) N_1}{(\sum_i p_i(\boldsymbol{x}) N_i)^2} \left(\frac{y - g(\boldsymbol{a}_1^\top \boldsymbol{x})}{\sigma^2} \right) g'(\boldsymbol{a}_1^\top \boldsymbol{x}) \boldsymbol{x}, & \text{if } i = 1 \\ \frac{-p_i(\boldsymbol{x}) p_1(\boldsymbol{x}) N_i N_1}{(\sum_i p_i(\boldsymbol{x}) N_i)^2} \left(\frac{y - g(\boldsymbol{a}_i^\top \boldsymbol{x})}{\sigma^2} \right) g'(\boldsymbol{a}_i^\top \boldsymbol{x}) \boldsymbol{x}, & \text{if } i \neq 1 \end{cases} \end{aligned}$$

Hence,

$$\mathbb{E}[(p^{(1)}(\boldsymbol{A},\boldsymbol{W})-p^{(1)}(\boldsymbol{A}^*,\boldsymbol{W}))\langle \boldsymbol{x},\widetilde{\Delta}\rangle] \tag{A.10}$$

$$=\sum_{i\in[k]}\int_0^1 \mathbb{E}[\langle \nabla_{\boldsymbol{a}_i} p^{(1)}(\boldsymbol{A}_t,\boldsymbol{W}),\Delta_i\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle]dt \tag{A.11}$$

$$=\int_0^1 \mathbb{E}\left[\frac{(\sum_{i\neq 1} p_i(\boldsymbol{x})N_i)p_1(\boldsymbol{x})N_1}{(\sum_i p_i(\boldsymbol{x})N_i)^2}\left(\frac{y-g(\boldsymbol{a}_1^\top \boldsymbol{x})}{\sigma^2}\right)g'(\boldsymbol{a}_1^\top \boldsymbol{x})\langle \boldsymbol{x},\Delta_1\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle\right]dt \tag{A.12}$$

$$+\sum_{i\neq 1}\int_0^1 \mathbb{E}\left[\frac{-p_i(\boldsymbol{x})p_1(\boldsymbol{x})N_iN_1}{(\sum_i p_i(\boldsymbol{x})N_i)^2}\left(\frac{y-g(\boldsymbol{a}_i^\top \boldsymbol{x})}{\sigma^2}\right)g'(\boldsymbol{a}_i^\top \boldsymbol{x})\langle \boldsymbol{x},\Delta_i\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle\right]dt, \tag{A.13}$$

where we denoted $(\boldsymbol{a}_i)_t$ by $\boldsymbol{a}_i$ in the integrals above(with a slight abuse of notation) for the sake of notational simplicity. For any $i\neq 1$, we have that

$$\begin{aligned}&\left|\frac{-p_i(\boldsymbol{x})p_1(\boldsymbol{x})N_iN_1}{(\sum_i p_i(\boldsymbol{x})N_i)^2}\left(\frac{y-g(\boldsymbol{a}_i^\top \boldsymbol{x})}{\sigma^2}\right)g'(\boldsymbol{a}_i^\top \boldsymbol{x})\langle \boldsymbol{x},\Delta_i\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle\right|\\ &\leq \frac{p_i(\boldsymbol{x})p_1(\boldsymbol{x})N_iN_1}{(p_1(\boldsymbol{x})N_1+p_i(\boldsymbol{x})N_i)^2}|(y-g(\boldsymbol{a}_i^\top \boldsymbol{x}))g'(\boldsymbol{a}_i^\top \boldsymbol{x})\langle \boldsymbol{x},\Delta_i/\sigma^2\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle|\end{aligned}$$

For g =linear, sigmoid and ReLU, we have that $|g'(\cdot)|\leq 1$. Moreover, $\frac{p_i(\boldsymbol{x})p_1(\boldsymbol{x})N_iN_1}{(p_1(\boldsymbol{x})N_1+p_i(\boldsymbol{x})N_i)^2}\leq 1/4$. Thus we have

$$\begin{aligned}&\frac{p_i(\boldsymbol{x})p_1(\boldsymbol{x})N_iN_1}{(p_1(\boldsymbol{x})N_1+p_i(\boldsymbol{x})N_i)^2}|(y-g(\boldsymbol{a}_i^\top \boldsymbol{x}))g'(\boldsymbol{a}_i^\top \boldsymbol{x})\langle \boldsymbol{x},\Delta_i/\sigma^2\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle|\\ &\leq \frac{1}{4}|y-g(\boldsymbol{a}_i^\top \boldsymbol{x})||\langle \boldsymbol{x},\Delta_i/\sigma^2\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle|.\end{aligned}$$

We thus get

$$\mathbb{E}\left[\frac{-p_i(\boldsymbol{x})p_1(\boldsymbol{x})N_iN_1}{(\sum_i p_i(\boldsymbol{x})N_i)^2}\left(\frac{y-g(\boldsymbol{a}_i^\top \boldsymbol{x})}{\sigma^2}\right)g'(\boldsymbol{a}_i^\top \boldsymbol{x})\langle \boldsymbol{x},\Delta_i\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle\right] \tag{A.14}$$

$$\leq \frac{1}{4}\mathbb{E}[|y-g(\boldsymbol{a}_i^\top \boldsymbol{x})||\langle \boldsymbol{x},\Delta_i/\sigma^2\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle|] \tag{A.15}$$

$$\leq \frac{1}{4}\sqrt{\mathbb{E}[(y-g(\boldsymbol{a}_i^\top \boldsymbol{x}))^2]\mathbb{E}[\langle \boldsymbol{x},\Delta_i/\sigma^2\rangle^2\langle \boldsymbol{x},\widetilde{\Delta}\rangle^2]} \tag{A.16}$$

$$\leq \frac{\sqrt{3}}{4}\sqrt{\mathbb{E}[(y-g(\boldsymbol{a}_i^\top \boldsymbol{x}))^2]}\|\Delta_i/\sigma^2\|_2\|\widetilde{\Delta}\|_2 \tag{A.17}$$

Now it remains to bound $\sqrt{\mathbb{E}[(y-g(\boldsymbol{a}_i^\top \boldsymbol{x}))^2]}$. Since $\|\boldsymbol{a}_i\|_2 \leq 1$, one can show that $\mathbb{E}[g(\boldsymbol{a}_i^\top \boldsymbol{x})^2] \leq 1$ for the given choice of non-linearities for g. Also, we have that

$$\begin{aligned}\mathbb{E}[y^2] = \mathbb{E}[\mathbb{E}[y^2|\boldsymbol{x}]] &= \mathbb{E}[\sum_{i\in[k]} p_i^*(\boldsymbol{x})g(\langle \boldsymbol{a}_i^*,\boldsymbol{x}\rangle)^2+\sigma^2] \\ &= \mathbb{E}[\sum_{i\in[k]} p_i^*(\boldsymbol{x})]\mathbb{E}[g(\langle \boldsymbol{a}_1^*,\boldsymbol{x}\rangle)^2]+\sigma^2 \leq 1+\sigma^2,\end{aligned}$$

where we used the following facts: (i) $\langle \boldsymbol{a}_i^*,\boldsymbol{x}\rangle$ is independent of the random variable $p_i^*(\boldsymbol{x})$ for each $i\in[k]$, (ii) $\langle \boldsymbol{a}_i^*,\boldsymbol{x}\rangle \overset{(d)}{=} \langle \boldsymbol{a}_1^*,\boldsymbol{x}\rangle$ and (iii) $\mathbb{E}[g(\langle \boldsymbol{a}_1^*,\boldsymbol{x}\rangle)^2] \leq 1$. Since $\mathbb{E}[(y-g(\boldsymbol{a}_i^\top \boldsymbol{x}))^2] \leq 2\mathbb{E}[y^2]+\mathbb{E}[g(\boldsymbol{a}_i^\top \boldsymbol{x})^2]$, after substituting these bounds in Equation A.17, we get

$$\begin{aligned}\mathbb{E}\left[\frac{-p_i(\boldsymbol{x})p_1(\boldsymbol{x})N_iN_1}{(\sum_i p_i(\boldsymbol{x})N_i)^2}\left(\frac{y-g(\boldsymbol{a}_i^\top \boldsymbol{x})}{\sigma^2}\right)g'(\boldsymbol{a}_i^\top \boldsymbol{x})\langle \boldsymbol{x},\Delta_i\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle\right]& \\ \leq \frac{\sqrt{6(2+\sigma^2)}}{4}\|\Delta_i/\sigma^2\|_2\|\widetilde{\Delta}\|_2.&\end{aligned}$$

Similarly,

$$\begin{aligned}\mathbb{E}\left[\frac{p_i(\boldsymbol{x})N_ip_1(\boldsymbol{x})N_1}{(\sum_i p_i(\boldsymbol{x})N_i)^2}\left(\frac{y-g(\boldsymbol{a}_1^\top \boldsymbol{x})}{\sigma^2}\right)g'(\boldsymbol{a}_1^\top \boldsymbol{x})\langle \boldsymbol{x},\Delta_1\rangle\langle \boldsymbol{x},\widetilde{\Delta}\rangle\right]& \\ \leq \frac{\sqrt{6(2+\sigma^2)}}{4}\|\Delta/\sigma^2\|_2\|\widetilde{\Delta}\|_2.&\end{aligned}$$

Substituting the above two inequalities in Equation A.13, we obtain that

$$\mathbb{E}[(p^{(1)}(\boldsymbol{A},\boldsymbol{W})-p^{(1)}(\boldsymbol{A}^*,\boldsymbol{W}))\langle \boldsymbol{x},\widetilde{\Delta}\rangle] \leq 2(k-1)\frac{\sqrt{6(2+\sigma^2)}}{4}\|\Delta_1/\sigma^2\|_2\|\widetilde{\Delta}\|_2.$$

Defining $\kappa \triangleq (k-1)\frac{\sqrt{6(2+\sigma^2)}}{2}$ and using the fact that $\|\Delta/\sigma^2\|_2 \leq \varepsilon_1$, we thus obtain

$$\|\mathbb{E}[(p^{(1)}(\boldsymbol{A}, \boldsymbol{W}) - p^{(1)}(\boldsymbol{A}^*, \boldsymbol{W}))\boldsymbol{x}]\|_2 \leq \kappa\varepsilon_1.$$

A.8 Proof of Lemma 7

A.9 Proof for $k = 2$

Proof. We first prove the lemma for $k = 2$. We show that the assumptions in Appendix A.1.1 hold *globally* in our setting yielding a geometric convergence. Here we simply denote $M(\boldsymbol{W}, \boldsymbol{A}^*)$ as $M(\boldsymbol{w})$ dropping the explicit dependence on $\boldsymbol{A}^*$. Recall that

$$Q(\boldsymbol{w}|\boldsymbol{w}_t) = \mathbb{E}_{p_{\boldsymbol{w}^*}(\boldsymbol{x},y)}\left[p_1(\boldsymbol{x}, y, \boldsymbol{w}_t) \cdot (\boldsymbol{w}^\top \boldsymbol{x}) - \log(1 + e^{\boldsymbol{w}^\top \boldsymbol{x}})\right],$$

where

$$p_1(\boldsymbol{x}, y, \boldsymbol{w}_t) = \frac{f(\boldsymbol{w}_t^\top \boldsymbol{x})\mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2)}{f(\boldsymbol{w}^\top \boldsymbol{x})\mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x}))\mathcal{N}(y|g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2)}. \tag{A.18}$$

For simplicity we drop the subscript in the above expectation with respect to the distribution $p_{\boldsymbol{w}^*}(\boldsymbol{x}, y)$. Now we verify each of the assumptions.

- Convexity of Ω easily follows from its definition.
- We have that

$$Q(\boldsymbol{w}|\boldsymbol{w}^*) = \mathbb{E}\left[p_1(\boldsymbol{x}, y, \boldsymbol{w}^*) \cdot (\boldsymbol{w}^\top \boldsymbol{x}) - \log(1 + e^{\boldsymbol{w}^\top \boldsymbol{x})}\right].$$

Note that the strong-concavity of $Q(\cdot|\boldsymbol{w}^*)$ is equivalent to the strong-convexity of $-Q(\cdot|\boldsymbol{w}^*)$. Denoting the sigmoid function by f, we have that for all $\boldsymbol{w} \in \Omega$,

$$\begin{aligned}
-\nabla^2 Q(\boldsymbol{w}|\boldsymbol{w}^*) &= \mathbb{E}\left[f'(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \boldsymbol{x}\boldsymbol{x}^\top\right], \\
&\overset{\text{(Stein's lemma)}}{=} \mathbb{E}\left[f'''(\boldsymbol{w}^\top \boldsymbol{x})\right] \cdot \boldsymbol{w}\boldsymbol{w}^\top + \mathbb{E}[f'(\boldsymbol{w}^\top \boldsymbol{x})] \cdot I \\
&= \mathbb{E}[f'''(\|\boldsymbol{w}\|Z)] \cdot \boldsymbol{w}\boldsymbol{w}^\top + \mathbb{E}[f'(\|\boldsymbol{w}\|Z)] \cdot I, \quad Z \sim \mathcal{N}(0,1) \\
&\overset{(a)}{\succcurlyeq} \inf_{0 \le \alpha \le 1} \min\left\{\mathbb{E}[f'(\alpha Z)], \mathbb{E}[f'(\alpha Z)] + \alpha^2 \mathbb{E}[f'''(\alpha Z)]\right\} \cdot I \\
&= \underbrace{0.14}_{\lambda} \cdot I
\end{aligned} \tag{A.19}$$

where (a) follows from finding the two possible eigenvalues of the positive-definite matrix in the previous step and considering the minimum among them to ensure strong-convexity. Here the value of λ is found numerically to be approximately around 0.1442.

- For any $\boldsymbol{w}, \boldsymbol{w}_t \in \Omega$,

$$\nabla Q(\boldsymbol{w}|\boldsymbol{w}_t) = \mathbb{E}\left[p_1(\boldsymbol{x}, y, \boldsymbol{w}_t) \cdot \boldsymbol{x} - f(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \boldsymbol{x}\right].$$

Thus,

$$\begin{aligned}
\|\nabla Q(M(\boldsymbol{w})|\boldsymbol{w}^*) - \nabla Q(M(\boldsymbol{w})|\boldsymbol{w})\| &= \|\mathbb{E}\left[(p_1(\boldsymbol{x}, y, \boldsymbol{w}_t) - p_1(\boldsymbol{x}, y, \boldsymbol{w}^*) \cdot \boldsymbol{x}\right]\| \\
&\overset{(a)}{\le} \gamma_\sigma \|\boldsymbol{w} - \boldsymbol{w}^*\|,
\end{aligned}$$

where we want to prove in (a) that γ_σ is smaller than 0.14 for all $\boldsymbol{w} \in \Omega$. Intuitively, this means that the posterior probability in Equation A.18 is smooth with respect to the parameter $\boldsymbol{w}$. We will now show that this can be achieved in the high-SNR regime when σ is sufficiently small. This will ensure that $\kappa_\sigma \triangleq \frac{\gamma_\sigma}{\lambda} < 1$. In particular, the value of γ_σ is dimension-independent and depends only on the choice of the non-linearity g.

To prove that

$$\|\mathbb{E}\left[(p_1(\boldsymbol{x}, y, \boldsymbol{w}) - p_1(\boldsymbol{x}, y, \boldsymbol{w}^*)) \cdot \boldsymbol{x}\right]\| \le \gamma \|\boldsymbol{w} - \boldsymbol{w}^*\| = \gamma \|\Delta\|,$$

it suffices to show

$$\langle \mathbb{E}\left[(p_1(\boldsymbol{x}, y, \boldsymbol{w}) - p_1(\boldsymbol{x}, y, \boldsymbol{w}^*)) \cdot \boldsymbol{x} \right], \widetilde{\Delta} \rangle \leq \gamma \|\Delta\| \|\widetilde{\Delta}\|, \quad \forall \widetilde{\Delta} \in \mathbb{R}^d.$$

Or equivalently,

$$\mathbb{E}\left[(p_1(\boldsymbol{x}, y, \boldsymbol{w}) - p_1(\boldsymbol{x}, y, \boldsymbol{w}^*)) \langle \boldsymbol{x}, \widetilde{\Delta} \rangle \right] \leq \gamma \|\Delta\| \|\widetilde{\Delta}\|.$$

Let $\Delta \triangleq \boldsymbol{w} - \boldsymbol{w}^*$ and $f(u) \triangleq p_1(\boldsymbol{x}, y, \boldsymbol{w}_u)$ where $\boldsymbol{w}_u = \boldsymbol{w}^* + u\Delta, u \in [0, 1]$. Thus $f(1) = p_1(\boldsymbol{x}, y, \boldsymbol{w})$ and $f(0) = p_1(\boldsymbol{x}, y, \boldsymbol{w}^*)$. So we get

$$\begin{aligned} p_1(\boldsymbol{x}, y, \boldsymbol{w}) - p_1(\boldsymbol{x}, y, \boldsymbol{w}^*) = f(1) - f(0) &= \int_0^1 f'(u) du \\ &= \int_0^1 \langle \nabla p_1(\boldsymbol{x}, y, \boldsymbol{w}_u), \Delta \rangle du, \end{aligned}$$

where the gradient is evaluated with respect to $\boldsymbol{w}_u$. Differentiating Equation A.18 with respect to $\boldsymbol{w}$, we get that

$$\begin{aligned} &\nabla_{\boldsymbol{w}} p_1(\boldsymbol{x}, y, \boldsymbol{w}) \\ &= \frac{f(\boldsymbol{w}^\top \boldsymbol{x})(1 - f(\boldsymbol{w}^\top \boldsymbol{x}))\mathcal{N}(y | g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) \mathcal{N}(y | g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2)}{(f(\boldsymbol{w}^\top \boldsymbol{x})\mathcal{N}(y | g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2) + (1 - f(\boldsymbol{w}^\top \boldsymbol{x}))\mathcal{N}(y | g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2))^2} \cdot \boldsymbol{x} \\ &\triangleq R(\boldsymbol{x}, y, \boldsymbol{w}, \sigma) \cdot \boldsymbol{x}. \end{aligned}$$

Thus,

$$\begin{aligned} &\mathbb{E}\left[(p_1(\boldsymbol{x}, y, \boldsymbol{w}) - p_1(\boldsymbol{x}, y, \boldsymbol{w}^*)) \langle \boldsymbol{x}, \widetilde{\Delta} \rangle \right] \\ &= \mathbb{E}\left[\left(\int_0^1 R(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma) \langle \boldsymbol{x}, \Delta \rangle du \right) \langle \boldsymbol{x}, \widetilde{\Delta} \rangle \right] \\ &= \int_0^1 \mathbb{E}\left[R(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma) \langle \boldsymbol{x}, \Delta \rangle \langle \boldsymbol{x}, \widetilde{\Delta} \rangle \right] du \\ &\leq \left(\int_0^1 \sqrt{\mathbb{E}[R(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2]} du \right) \sqrt{\mathbb{E}\left[\langle \boldsymbol{x}, \Delta \rangle^2 \langle \boldsymbol{x}, \widetilde{\Delta} \rangle^2 \right]} \\ &\leq \underbrace{\sqrt{3} \left(\int_0^1 \sqrt{\mathbb{E}[R(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2]} du \right)}_{\gamma_\sigma} \|\Delta\| \|\widetilde{\Delta}\| \\ &= \gamma_\sigma \|\Delta\| \|\widetilde{\Delta}\|, \end{aligned}$$

where the last inequality follows from Lemma 5 of [27]. Our goal is to now prove that $\gamma_\sigma \to 0$ as $\sigma \to 0$. First observe that

$$\begin{aligned}
R(\boldsymbol{x}, y, \boldsymbol{w}, \sigma) &= \frac{f(\boldsymbol{w}^\top \boldsymbol{x})(1-f(\boldsymbol{w}^\top \boldsymbol{x})e^{-(y-g(\boldsymbol{a}_1^\top \boldsymbol{x}))/2\sigma^2} e^{-(y-g(\boldsymbol{a}_1^\top \boldsymbol{x}))/2\sigma^2}}{(f(\boldsymbol{w}^\top \boldsymbol{x})e^{-(y-g(\boldsymbol{a}_1^\top \boldsymbol{x}))/2\sigma^2} + (1-f(\boldsymbol{w}^\top \boldsymbol{x}))e^{-(y-g(\boldsymbol{a}_2^\top \boldsymbol{x}))/2\sigma^2})^2} \\
&= \frac{f(1-f)e^{\frac{(y-g(\boldsymbol{a}_1^\top \boldsymbol{x}))^2-(y-g(\boldsymbol{a}_2^\top \boldsymbol{x}))^2}{2\sigma^2}}}{\left(f+(1-f)e^{\frac{(y-g(\boldsymbol{a}_1^\top \boldsymbol{x}))^2-(y-g(\boldsymbol{a}_2^\top \boldsymbol{x}))^2}{2\sigma^2}}\right)^2} \leq 1/4 \\
&\to 0 \text{ as } \sigma \to 0,
\end{aligned}$$

where the key observation is that irrespective of the sign of $(y - g(\boldsymbol{a}_1^\top \boldsymbol{x}))^2 - (y - g(\boldsymbol{a}_2^\top \boldsymbol{x}))^2$, the ratio still goes to zero and hence by dominated convergence theorem $\mathbb{E}[R(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2] \to 0$ for each $u \in [0,1]$. Now we show that this convergence is uniform in u and thus $\gamma_\sigma \to 0$. For simplicity, define

$$\Delta_1 \triangleq (y - g(\boldsymbol{a}_1^\top \boldsymbol{x}))^2, \quad \Delta_2 \triangleq (y - g(\boldsymbol{a}_2^\top \boldsymbol{x}))^2 \text{ and } \sigma = \frac{1}{n}. \tag{A.20}$$

Thus,

$$\begin{aligned}
R(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma) &= \frac{f(1-f)e^{\frac{n^2}{2}(\Delta_1-\Delta_2)}}{\left(f+(1-f)e^{\frac{n^2}{2}(\Delta_1-\Delta_2)}\right)^2} && \text{(A.21)} \\
&\leq \frac{f(1-f)e^{\frac{n^2}{2}(\Delta_1-\Delta_2)}}{\left((1-f)e^{\frac{n^2}{2}(\Delta_1-\Delta_2)}\right)^2} = \frac{f}{1-f}e^{-\frac{n^2}{2}(\Delta_1-\Delta_2)}. && \text{(A.22)}
\end{aligned}$$

Similarly,

$$R(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma) \leq \frac{1-f}{f}e^{-\frac{n^2}{2}(\Delta_2-\Delta_1)}. \tag{A.23}$$

Thus, we get

$$R(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma) \leq \max\left(\frac{1-f}{f}, \frac{f}{1-f}\right) e^{-\frac{n^2}{2}(|\Delta_1-\Delta_2|)}. \tag{A.24}$$

Hence

$$\frac{\gamma_\sigma}{\sqrt{3}} = \int_0^1 \sqrt{\mathbb{E}[\text{Ratio}(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2]}du \tag{A.25}$$

$$\leq \int_0^1 \sqrt{\mathbb{E}\left[\max\left(\frac{1-f}{f}, \frac{f}{1-f}\right)^2 e^{-n^2|\Delta_1-\Delta_2|}\right]}du \tag{A.26}$$

$$\leq \int_0^1 \sqrt{\mathbb{E}\left[\left(\frac{1-f}{f}\right)^2 e^{-n^2|\Delta_1-\Delta_2|} + \left(\frac{f}{1-f}\right)^2 e^{-n^2|\Delta_1-\Delta_2|}\right]}du \tag{A.27}$$

$$= \int_0^1 \sqrt{2\mathbb{E}\left[e^{2\boldsymbol{w}_u^\top \boldsymbol{x}} e^{-n^2|\Delta_1-\Delta_2|}\right]}du \tag{A.28}$$

$$\leq \int_0^1 \sqrt{2\sqrt{\mathbb{E}[e^{4\boldsymbol{w}_u^\top \boldsymbol{x}}]\mathbb{E}[e^{-2n^2|\Delta_1-\Delta_2|}]}}du \tag{A.29}$$

$$\overset{(a)}{\leq} \sqrt{2e^4\sqrt{\mathbb{E}[e^{-2n^2|\Delta_1-\Delta_2|}]}}, \tag{A.30}$$

where (a) follows from the fact $\|\boldsymbol{w}_u\| \leq 1$ and $\mathbb{E}[e^{4\boldsymbol{w}_u^\top \boldsymbol{x}}] = e^{8\|\boldsymbol{w}_u\|^2} \leq e^8$, for each $u \in [0, 1]$. Now we analyze the convergence rate of the last term $\mathbb{E}[e^{-2n^2|\Delta_1-\Delta_2|}]$ for the case of linear regression, i.e. $g(z) = z$. Notice that for the two-mixtures, we have

$$y \overset{(d)}{=} Z(\boldsymbol{a}_1^\top \boldsymbol{x}) + (1-Z)\boldsymbol{a}_2^\top \boldsymbol{x} + \sigma N = Z(\boldsymbol{a}_1^\top \boldsymbol{x}) + (1-Z)\boldsymbol{a}_2^\top \boldsymbol{x} + \frac{N}{n}, \tag{A.31}$$

$$Z|\boldsymbol{x} \sim \text{Bern}(f(\boldsymbol{w}_*^\top \boldsymbol{x})). \tag{A.32}$$

Thus,

$$\Delta_1 - \Delta_2 \overset{(d)}{=} (y - \boldsymbol{a}_1^\top \boldsymbol{x})^2 - (y - \boldsymbol{a}_2^\top \boldsymbol{x})^2 \tag{A.33}$$

$$= (\boldsymbol{a}_1^\top \boldsymbol{x} - \boldsymbol{a}_2^\top \boldsymbol{x})^2(1-2Z) + \frac{2N}{n}(\boldsymbol{a}_2^\top \boldsymbol{x} - \boldsymbol{a}_1^\top \boldsymbol{x}) \tag{A.34}$$

$$= \langle \boldsymbol{x}, \boldsymbol{v}\rangle^2(1-2Z) + \frac{2N}{n}\langle \boldsymbol{x}, \boldsymbol{v}\rangle, \quad \boldsymbol{v} = \boldsymbol{a}_1 - \boldsymbol{a}_2. \tag{A.35}$$

Since Z can equal either 0 or 1, we have

$$
\begin{aligned}
\gamma_\sigma &\leq \sqrt{3}\sqrt{2e^4}\left(\mathbb{E}[e^{-2n^2\left|\langle \boldsymbol{x},\boldsymbol{v}\rangle^2(1-2Z)+\frac{2N}{n}\langle \boldsymbol{x},\boldsymbol{v}\rangle\right|}]\right)^{1/4} && \text{(A.36)}\\
&\leq \sqrt{6e^4}\left(\mathbb{E}\left[\max\left(e^{-2n^2\left|\langle \boldsymbol{x},\boldsymbol{v}\rangle^2+\frac{2N}{n}\langle \boldsymbol{x},\boldsymbol{v}\rangle\right|}, e^{-2n^2\left|-\langle \boldsymbol{x},\boldsymbol{v}\rangle^2+\frac{2N}{n}\langle \boldsymbol{x},\boldsymbol{v}\rangle\right|}\right)\right]\right)^{1/4} && \text{(A.37)}\\
&\leq \sqrt{6\sqrt{2}e^4}\left(\mathbb{E}\left[e^{-2n^2|\langle \boldsymbol{x},\boldsymbol{v}\rangle^2+\frac{2N}{n}\langle \boldsymbol{x},\boldsymbol{v}\rangle|}\right]\right)^{1/4} && \text{(A.38)}\\
&= \sqrt{6\sqrt{2}e^4}\left(\mathbb{E}\left[e^{-2n^2|Z^2+\frac{2ZN}{n}|}\right]\right)^{1/4}, \quad Z\sim\mathcal{N}(0,\|\boldsymbol{a}_1-\boldsymbol{a}_2\|), N\sim\mathcal{N}(0,1). && \text{(A.39)}\\
&= O\left(\sqrt{6\sqrt{2}e^4}\left(\mathbb{E}[e^{-2n^2Z^2}]\right)^{1/4}\right) && \text{(A.40)}\\
&= \sqrt{6\sqrt{2}e^4}\left(\sqrt{\frac{1}{4n^2\|\boldsymbol{a}_1-\boldsymbol{a}_2\|^2+1}}\right)^{1/4} && \text{(A.41)}\\
&= O\left(\frac{1}{(n\|\boldsymbol{a}_1-\boldsymbol{a}_2\|)^{1/4}}\right) && \text{(A.42)}\\
&= O\left(\left(\frac{\sigma}{\|\boldsymbol{a}_1-\boldsymbol{a}_2\|}\right)^{1/4}\right). && \text{(A.43)}
\end{aligned}
$$

□

A.10 Proof for general k

Proof. The proof strategy for general k is similar. First let $\varepsilon_1 = 0$. Our task is to show that the assumptions of Appendix A.1.1 hold globally in our setting. The domain Ω is clearly convex since

$$\Omega = \{\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{k-1}) : \|\boldsymbol{w}_i\| \leq 1, \forall i \in [k-1]\}.$$

Now we verify Assumption 2. The function $Q(.|\boldsymbol{w}_t)$ is given by

$$Q(\boldsymbol{w}|\boldsymbol{w}_t) = \mathbb{E}\left[\sum_{i\in[k-1]} p^{(i)}_{\boldsymbol{w}_t}(\boldsymbol{w}_i^\top \boldsymbol{x}) - \log\left(1 + \sum_{i\in[k-1]} e^{\boldsymbol{w}_i^\top \boldsymbol{x}}\right)\right],$$

where $p^{(i)}_{\boldsymbol{w}_t} \triangleq \mathbb{P}\left(z = i | \boldsymbol{x}, y, \boldsymbol{w}_t\right)$ corresponds to the posterior probability for the i^{th} expert, given by

$$p^{(i)}_{\boldsymbol{w}_t} = \frac{p_{i,t}(\boldsymbol{x}) \mathcal{N}(y | g(\boldsymbol{a}_i^\top \boldsymbol{x}), \sigma^2)}{\sum_{j \in [k]} p_{j,t}(\boldsymbol{x}) \mathcal{N}(y | g(\boldsymbol{a}_j^\top \boldsymbol{x}), \sigma^2)}, \quad p_{i,t}(\boldsymbol{x}) = \frac{e^{(\boldsymbol{w}_t)_i^\top \boldsymbol{x}}}{1 + \sum_{j \in [k-1]} e^{(\boldsymbol{w}_t)_j^\top \boldsymbol{x}}}.$$

Throughout we follow the convention that $\boldsymbol{w}_k = 0$. Thus the gradient of Q with respect to the i^{th} gating parameter $\boldsymbol{w}_i$ is given by

$$\nabla_{\boldsymbol{w}_i} Q(\boldsymbol{w} | \boldsymbol{w}_t) = \mathbb{E}\left[\left(p^{(i)}_{\boldsymbol{w}_t} - \frac{e^{\boldsymbol{w}_i^\top \boldsymbol{x}}}{1 + \sum_{j \in [k-1]} e^{\boldsymbol{w}_j^\top \boldsymbol{x}}}\right) \cdot \boldsymbol{x}\right], \quad i \in [k-1].$$

Thus the $(i,j)^{\text{th}}$ block of the negative Hessian $-\nabla^{(2)}_{\boldsymbol{w}} Q(\boldsymbol{w} | \boldsymbol{w}^*) \in \mathbb{R}^{d(k-1) \times d(k-1)}$ is given by

$$-\nabla_{\boldsymbol{w}_i, \boldsymbol{w}_j} Q(\boldsymbol{w} | \boldsymbol{w}^*) = \begin{cases} \mathbb{E}[p_i(\boldsymbol{x})(1 - p_i(\boldsymbol{x})) \cdot \boldsymbol{x}\boldsymbol{x}^\top], & j = i \\ \mathbb{E}[-p_i(\boldsymbol{x}) p_j(\boldsymbol{x}) \cdot \boldsymbol{x}\boldsymbol{x}^\top], & j \neq i \end{cases}, \tag{A.44}$$

where $p_i(\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_i^\top \boldsymbol{x}}}{1 + \sum_{j \in [k-1]} e^{\boldsymbol{w}_j^\top \boldsymbol{x}}}$. It is clear from Equation A.44 that $-\nabla^{(2)}_{\boldsymbol{w}} Q(\boldsymbol{w} | \boldsymbol{w}^*)$ is positive semi-definite. Since we are interested in the strong convexity of $-Q(\cdot | \boldsymbol{w}^*)$ which is equivalent to positive definiteness of the negative Hessian, it suffices to show that

$$\lambda \triangleq \inf_{w \in \Omega} \lambda_{\min}\left(-\nabla^{(2)}_{\boldsymbol{w}} Q(\boldsymbol{w} | \boldsymbol{w}^*)\right) > 0.$$

Since the Hessian is continuous with respect to $\boldsymbol{w}$ and consequently the minimum eigenvalue of it, there exists a $\boldsymbol{w}' \in \Omega$ such that

$$\lambda = \lambda_{\min}\left(-\nabla^{(2)}_{\boldsymbol{w}'} Q(\boldsymbol{w}' | \boldsymbol{w}^*)\right) = \inf_{\|\boldsymbol{a}\| = 1} \boldsymbol{a}^\top \left(-\nabla^{(2)}_{\boldsymbol{w}'} Q(\boldsymbol{w}' | \boldsymbol{w}^*)\right) \boldsymbol{a},$$

where $\boldsymbol{a} = (\boldsymbol{a}_1^\top, \ldots, \boldsymbol{a}_{k-1}^\top)^\top \in \mathbb{R}^{d(k-1)}$. In view of Equation A.44, the above equation can be further simplified to

$$\lambda = \inf_{\|\boldsymbol{a}\| = 1} \mathbb{E}[\boldsymbol{a}_{\boldsymbol{x}}^\top M_{\boldsymbol{x}} \boldsymbol{a}_{\boldsymbol{x}}], \tag{A.45}$$

where $\boldsymbol{a}_{\boldsymbol{x}} = (\boldsymbol{a}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{a}_{k-1}^\top \boldsymbol{x})^\top \in \mathbb{R}^{k-1}$ and $M_{\boldsymbol{x}}$ is given by

$$M_{\boldsymbol{x}}(i,j) = \begin{cases} p_i(\boldsymbol{x})(1 - p_i(\boldsymbol{x})), & i = j \\ -p_i(\boldsymbol{x}) p_j(\boldsymbol{x}), & i \neq j \end{cases}$$

Let the infimum in Equation A.45 is attained by $\boldsymbol{a}^*$, i.e. $\lambda = \mathbb{E}[(\boldsymbol{a}_{\boldsymbol{x}}^*)^\top M_{\boldsymbol{x}} \boldsymbol{a}_{\boldsymbol{x}}^*]$. For each $\boldsymbol{x}$, $M_{\boldsymbol{x}}$ is strictly diagonally dominant since $|M_{\boldsymbol{x}}(i,i)| = p_i(\boldsymbol{x})(1 - p_i(\boldsymbol{x})) = p_i(\boldsymbol{x}) \left(\sum_{j \neq i, j \in [k]} p_j(\boldsymbol{x}) \right) > p_i(\boldsymbol{x}) \left(\sum_{j \neq i, j \in [k-1]} p_j(\boldsymbol{x}) \right) = \sum_{j \neq i} M(i,j)$. Thus $M_{\boldsymbol{x}}$ is positive-definite and $(\boldsymbol{a}_{\boldsymbol{x}}^*)^\top M_{\boldsymbol{x}} \boldsymbol{a}_{\boldsymbol{x}}^* > 0$ whenever $\boldsymbol{a}_{\boldsymbol{x}}^* \neq 0$. Since x follows a continuous distribution it follows that $\boldsymbol{a}_{\boldsymbol{x}}^* \neq 0$ with probability 1 and thus $\lambda = \mathbb{E}[(\boldsymbol{a}_{\boldsymbol{x}}^*)^\top M_{\boldsymbol{x}} \boldsymbol{a}_{\boldsymbol{x}}^*] > 0$.

Now it remains to show that Assumption 3 too holds, i.e.

$$\|\nabla Q(M(\boldsymbol{w})|\boldsymbol{w}^*) - \nabla Q(M(\boldsymbol{w})|\boldsymbol{w})\| \leq \gamma \|\boldsymbol{w} - \boldsymbol{w}^*\|.$$

Note that $\boldsymbol{w} = (\boldsymbol{w}_1^\top, \ldots, \boldsymbol{w}_{k-1}^\top)^\top \in \mathbb{R}^{d(k-1)}$. We will show that

$$\|(\nabla Q(M(\boldsymbol{w})|\boldsymbol{w}^*))_i - (\nabla Q(M(\boldsymbol{w})|\boldsymbol{w}))_i\| \leq \gamma_\sigma \|\boldsymbol{w} - \boldsymbol{w}^*\|, \quad i \in [k-1],$$

where $(\nabla Q(M(\boldsymbol{w})|\boldsymbol{w}))_i \in \mathbb{R}^d$ refers to the i^{th} block of the gradient and $\gamma_\sigma \to 0$. Observe that

$$(\nabla Q(M(\boldsymbol{w})|\boldsymbol{w}^*))_i - (\nabla Q(M(\boldsymbol{w})|\boldsymbol{w}))_i = \mathbb{E}\left[(p_{\boldsymbol{w}}^{(i)} - p_{\boldsymbol{w}^*}^{(i)}) \cdot \boldsymbol{x} \right]$$

Let $\Delta = \boldsymbol{w} - \boldsymbol{w}^*$ and correspondingly $\Delta = (\Delta_1^\top, \ldots, \Delta_{k-1}^\top)^\top$ where $\Delta_i = \boldsymbol{w}_i - \boldsymbol{w}_i^*$. Thus it suffices to show that

$$\|\mathbb{E}[(p_{\boldsymbol{w}}^{(i)} - p_{\boldsymbol{w}^*}^{(i)}) \cdot \boldsymbol{x}]\| \leq \gamma_\sigma \|\Delta\|.$$

Or equivalently,

$$\mathbb{E}[(p_{\boldsymbol{w}}^{(i)} - p_{\boldsymbol{w}^*}^{(i)}) \langle \boldsymbol{x}, \widetilde{\Delta} \rangle] \leq \gamma_\sigma \|\Delta\| \|\widetilde{\Delta}\|, \quad \forall \widetilde{\Delta} \in \mathbb{R}^d.$$

We consider the case $i = 1$. The proof for the other cases is similar. Recall that

$$p_{\boldsymbol{w}}^{(1)} = \frac{p_1(\boldsymbol{x})\mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2)}{\sum_{j\in[k]} p_j(\boldsymbol{x})\mathcal{N}(y|g(\boldsymbol{a}_j^\top \boldsymbol{x}), \sigma^2)}, p_i(\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_i^\top \boldsymbol{x}}}{1 + \sum_{j\in[k-1]} e^{\boldsymbol{w}_j^\top \boldsymbol{x}}}, i \in [k-1].$$

For simplicity we define $N_i = \mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2)$. It is straightforward to verify that

$$\nabla_{\boldsymbol{w}_j} p_i(\boldsymbol{x}) = \begin{cases} p_i(\boldsymbol{x})(1 - p_i(\boldsymbol{x})) \cdot \boldsymbol{x}, & j = i \\ -p_i(\boldsymbol{x})p_j(\boldsymbol{x}) \cdot \boldsymbol{x}, & j \neq i \end{cases}$$

Thus

$$\begin{aligned}
\nabla_{\boldsymbol{w}_1}(p_{\boldsymbol{w}}^{(1)}) &= \nabla_{\boldsymbol{w}_1}\left(\frac{p_1(\boldsymbol{x})N_1}{\sum_{i=1}^N p_i(\boldsymbol{x})N_i}\right) \\
&= \frac{\boldsymbol{x}}{\left(\sum_{i=1}^N p_i(\boldsymbol{x})N_i\right)^2}((\sum_{i=1}^N p_i(\boldsymbol{x})N_i)p_1(\boldsymbol{x})(1 - p_1(\boldsymbol{x}))N_1 \\
&- p_1(\boldsymbol{x})N_1(-\sum_{j\neq 1} p_j(\boldsymbol{x})p_1(\boldsymbol{x})N_j + p_1(\boldsymbol{x})(1 - p_1(\boldsymbol{x}))N_1)) \\
&= \frac{p_1(\boldsymbol{x})N_1\left(\sum_{j\geq 2} p_j(\boldsymbol{x})N_j\right)}{\left(\sum_{i=1}^N p_i(\boldsymbol{x})N_i\right)^2} \cdot \boldsymbol{x} \\
&\triangleq R_1(\boldsymbol{x}, y, \boldsymbol{w}, \sigma) \cdot \boldsymbol{x}
\end{aligned}$$

Similarly,

$$\begin{aligned}
\nabla_{\boldsymbol{w}_i}(p_{\boldsymbol{w}}^{(1)}) &= \frac{p_1(\boldsymbol{x})p_i(\boldsymbol{x})N_1 N_i}{\left(\sum_{i=1}^N p_i(\boldsymbol{x})N_i\right)^2} \cdot \boldsymbol{x}, \quad i \neq 1, \\
&\triangleq R_i(\boldsymbol{x}, y, \boldsymbol{w}, \sigma) \cdot \boldsymbol{x}.
\end{aligned}$$

Let $\boldsymbol{w}_u \triangleq \boldsymbol{w}^* + u\Delta, u \in [0,1]$ and $f(u) \triangleq p^{(1)}_{\boldsymbol{w}_u}$. Thus

$$\begin{aligned}
p^{(1)}_{\boldsymbol{w}} - p^{(1)}_{\boldsymbol{w}^*} = f(1) - f(0) &= \int_0^1 f'(u)du \\
&= \int_0^1 \left(\sum_{i\in[k-1]} \langle \nabla_{\boldsymbol{w}_i}(p^{(1)}_{\boldsymbol{w}_u}), \Delta_i\rangle \right) du \\
&= \sum_{i\in[k-1]} \int_0^1 R_i(\boldsymbol{x}, y, \boldsymbol{w}, \sigma)\langle \boldsymbol{x}, \Delta_i\rangle du.
\end{aligned}$$

So we get

$$\begin{aligned}
\mathbb{E}[(p^{(1)}_{\boldsymbol{w}} - p^{(1)}_{\boldsymbol{w}^*})\langle \boldsymbol{x}, \widetilde{\Delta}\rangle] &= \sum_{i\in[k-1]} \int_0^1 \mathbb{E}[R_i(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)\langle \boldsymbol{x}, \Delta_i\rangle\langle \boldsymbol{x}, \widetilde{\Delta}\rangle]du \\
&\le \sum_{i\in[k-1]} \int_0^1 \sqrt{\mathbb{E}[R_i(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2]\mathbb{E}[\langle \boldsymbol{x}, \Delta_i\rangle^2\langle \boldsymbol{x}, \widetilde{\Delta}\rangle^2]}du \\
&\le \sum_{i\in[k-1]} \int_0^1 \sqrt{\mathbb{E}[R_i(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2]} \left(\sqrt{3}\|\Delta_i\|\|\widetilde{\Delta}\|\right) du \\
&\le \sum_{i\in[k-1]} \int_0^1 \sqrt{\mathbb{E}[R_i(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2]} \left(\sqrt{3}\|\Delta\|\|\widetilde{\Delta}\|\right) du \\
&= \underbrace{\left(\sum_{i\in[k-1]} \int_0^1 \sqrt{\mathbb{E}[R_i(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2]}du \right)}_{\gamma^{(1)}_\sigma} \left(\sqrt{3}\|\Delta\|\|\widetilde{\Delta}\|\right)
\end{aligned}$$

Now our goal is to show that $\mathbb{E}[R_i(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2] \to 0$ as $\sigma \to 0$. For $i = 1$, we have

$$\begin{aligned}
R_1(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2 = \left(\frac{\sum_{j\ge 2} p_1(\boldsymbol{x})p_j(\boldsymbol{x})N_1N_j}{\left(\sum_{i=1}^N p_i(\boldsymbol{x})N_i\right)^2} \right)^2 &\le k\sum_{j\ge 2} \left(\frac{p_1(\boldsymbol{x})p_j(\boldsymbol{x})N_1N_j}{\left(\sum_{i=1}^N p_i(\boldsymbol{x})N_i\right)^2} \right)^2 \\
\le k\sum_{j\ge 2} \left(\frac{p_1(\boldsymbol{x})p_j(\boldsymbol{x})N_1N_j}{(p_1(\boldsymbol{x})N_1 + p_j(\boldsymbol{x})N_j)^2} \right)^2
\end{aligned}$$

Similarly,

$$R_i(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2 \le \left(\frac{p_1(\boldsymbol{x})p_i(\boldsymbol{x})N_1N_i}{(p_1(\boldsymbol{x})N_1 + p_i(\boldsymbol{x})N_i)^2} \right)^2, \quad \forall i \ne 1, i \in [k-1].$$

For $\boldsymbol{w} = \boldsymbol{w}_u$ and $i \neq 1$, we have that

$$\begin{aligned}
\frac{p_1(\boldsymbol{x})p_i(\boldsymbol{x})N_1N_i}{(p_1(\boldsymbol{x})N_1 + p_i(\boldsymbol{x})N_i)^2} &= \frac{e^{\boldsymbol{w}_1^\top \boldsymbol{x}} e^{\boldsymbol{w}_i^\top \boldsymbol{x}} e^{-\frac{(y-g(\boldsymbol{a}_1^\top \boldsymbol{x}))^2}{2\sigma^2}} e^{-\frac{(y-g(\boldsymbol{a}_i^\top \boldsymbol{x}))^2}{2\sigma^2}}}{\left(e^{\boldsymbol{w}_1^\top \boldsymbol{x}} e^{-\frac{(y-g(\boldsymbol{a}_1^\top \boldsymbol{x}))^2}{2\sigma^2}} + e^{\boldsymbol{w}_i^\top \boldsymbol{x}} e^{-\frac{(y-g(\boldsymbol{a}_i^\top \boldsymbol{x}))^2}{2\sigma^2}}\right)^2} \leq \frac{1}{4} \\
&= \frac{e^{\boldsymbol{w}_1^\top \boldsymbol{x}} e^{\boldsymbol{w}_i^\top \boldsymbol{x}} e^{\frac{(y-g(\boldsymbol{a}_1^\top \boldsymbol{x}))^2-(y-g(\boldsymbol{a}_i^\top \boldsymbol{x}))^2}{2\sigma^2}}}{\left(e^{\boldsymbol{w}_1^\top \boldsymbol{x}} + e^{\boldsymbol{w}_i^\top \boldsymbol{x}} e^{\frac{(y-g(\boldsymbol{a}_1^\top \boldsymbol{x}))^2-(y-g(\boldsymbol{a}_i^\top \boldsymbol{x}))^2}{2\sigma^2}}\right)^2} \\
&\xrightarrow{\sigma \to 0} 0.
\end{aligned}$$

Thus, by Dominated Convergence Theorem, $\mathbb{E}[R_i(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2] \to 0$ for each $u \in [0, 1]$. To show that $\int_0^1 \mathbb{E}[R_i(\boldsymbol{x}, y, \boldsymbol{w}_u, \sigma)^2]du \to 0$, we can now follow the same analysis as in the proof of Theorem 2 from Equation A.20 onwards (replacing $\boldsymbol{w}$ there with $\boldsymbol{w}_1 - \boldsymbol{w}_i$) which ensures that $\gamma_\sigma^{(1)}$ in our case converges to zero. Similarly for other $i \in [k-1]$, we get that $\gamma^{(i)} \to 0$. Taking $\gamma_\sigma = \gamma_\sigma^{(1)} + \ldots + \gamma_\sigma^{(k-1)}$ and $\kappa_\sigma = \frac{\gamma_\sigma}{\lambda}$ completes the proof.

□

A.11 Gradient EM algorithm

In this section, we provide the convergence guarantees for the gradient EM algorithm. For simplicity, we prove the results for $k = 2$ and $(\boldsymbol{a}_1, \boldsymbol{a}_2) = (\boldsymbol{a}_1^*, \boldsymbol{a}_2^*)$. Thus we want to learn the gating parameter $\boldsymbol{w}^*$ in this setting. The results for the general case follow essentially the same proof as that of Theorem 2. In particular, our Theorem 8 can be viewed as a generalization of Lemma 7. Together with Lemma 8, extension to general k is straightforward.

Note that in the M-step of the EM algorithm, instead of maximizing $Q(\cdot|\boldsymbol{w}_t)$, we can chose an iterate so that it increases the Q value instead of fully maximizing it, i.e. $Q(\boldsymbol{w}_{t+1}|\boldsymbol{w}_t) \geq Q(\boldsymbol{w}_t|\boldsymbol{w}_t)$. Such a procedure is termed as *generalized EM*. *Gradient EM* is an example of generalized EM in which we take an ascent step in the direction of the gradient of $Q(\cdot|\boldsymbol{w}_t)$ to produce the next iterate, i.e.

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \nabla Q(\boldsymbol{w}_t|\boldsymbol{w}_t),$$

where $\alpha > 0$ is a suitably chosen step size and the gradient is with respect to the first argument. To account for the constrained optimization, we can include a projection step. Mathematically,

$$\boldsymbol{w}_{t+1} = G(\boldsymbol{w}_t), \quad G(\boldsymbol{w}) = \Pi_\Omega(\boldsymbol{w} + \alpha \nabla Q(\boldsymbol{w}|)\boldsymbol{w}),$$

where Π_Ω refers to the projection operator. Our next result establishes that the iterates of the gradient EM algorithm too converge geometrically for an appropriately chosen step size α.

Theorem 8. *Suppose that the domain $\Omega = \{\boldsymbol{w} \in \mathbb{R}^d : \|\boldsymbol{w}\|_2 \leq 1\}$ and $(\boldsymbol{a}_1, \boldsymbol{a}_2) = (\boldsymbol{a}_1^*, \boldsymbol{a}_2^*)$. Then there exist constants $\alpha_0 > 0$ and $\sigma_0 > 0$ such that for any step size $0 < \alpha \leq \alpha_0$ and noise variance $\sigma < \sigma_0$, the gradient EM updates on the gating parameter $\{\boldsymbol{w}\}_{t \geq 0}$ converge geometrically to the true parameter $\boldsymbol{w}^*$, i.e.*

$$\|\boldsymbol{w}_t - \boldsymbol{w}^*\| \leq (\rho_\sigma)^t \, \|\boldsymbol{w}_0 - \boldsymbol{w}^*\|,$$

where ρ_σ is a dimension-independent constant depending on g and σ.

Remark 4. The condition $\sigma < \sigma_0$ ensures that the Lipschitz constant ρ_σ for the map G is strictly less than 1. The constant α_0 depends only on two universal constants which are nothing but the strong-concavity and the smoothness parameters for the function $Q(\cdot|\boldsymbol{w}^*)$.

Proof. In addition to the assumptions of Appendix A.1.1, if we can ensure that the map $-Q(\cdot|\boldsymbol{w}^*)$ is μ-smooth, then the proof follows from Theorem 3 of [27] if we choose $\alpha_0 = \frac{2}{\mu+\lambda}$ where λ is the strong-convexity parameter of $-Q(\cdot|\boldsymbol{w}^*)$. The strong-convexity is already established in Appendix A.6. To find the smoothness parameter, note that

$$\begin{aligned}
-\nabla^2 Q(\boldsymbol{w}|\boldsymbol{w}^*) &= \mathbb{E}\left[f'(\boldsymbol{w}^\top \boldsymbol{x}) \cdot \boldsymbol{x}\boldsymbol{x}^\top\right], \\
&= \mathbb{E}\left[f'''(\boldsymbol{w}^\top \boldsymbol{x})\right] \cdot \boldsymbol{w}\boldsymbol{w}^\top + \mathbb{E}[f'(\boldsymbol{w}^\top \boldsymbol{x})] \cdot I \\
&= \mathbb{E}[f'''(\|\boldsymbol{w}\|Z)] \cdot \boldsymbol{w}\boldsymbol{w}^\top + \mathbb{E}[f'(\|\boldsymbol{w}\|Z)] \cdot I, \quad Z \sim \mathcal{N}(0,1) \\
&\preceq \sup_{0 \leq \alpha \leq 1} \min\left\{\mathbb{E}[f'(\alpha Z)], \mathbb{E}[f'(\alpha Z)] + \alpha^2 \mathbb{E}[f'''(\alpha Z)]\right\} \cdot I \\
&= \underbrace{0.25}_{\mu} \cdot I.
\end{aligned}$$

The contraction parameter is then given by

$$\rho_\sigma = 1 - \frac{2\lambda + 2\gamma_\sigma}{\mu + \lambda}.$$

Since $\gamma_\sigma \xrightarrow{\sigma \to 0} 0$, $\rho_\sigma < 1$ whenever $\sigma < \sigma_0$ for a constant σ_0. $\square$

A.12 Additional experiments

A.12.1 Synthetic data

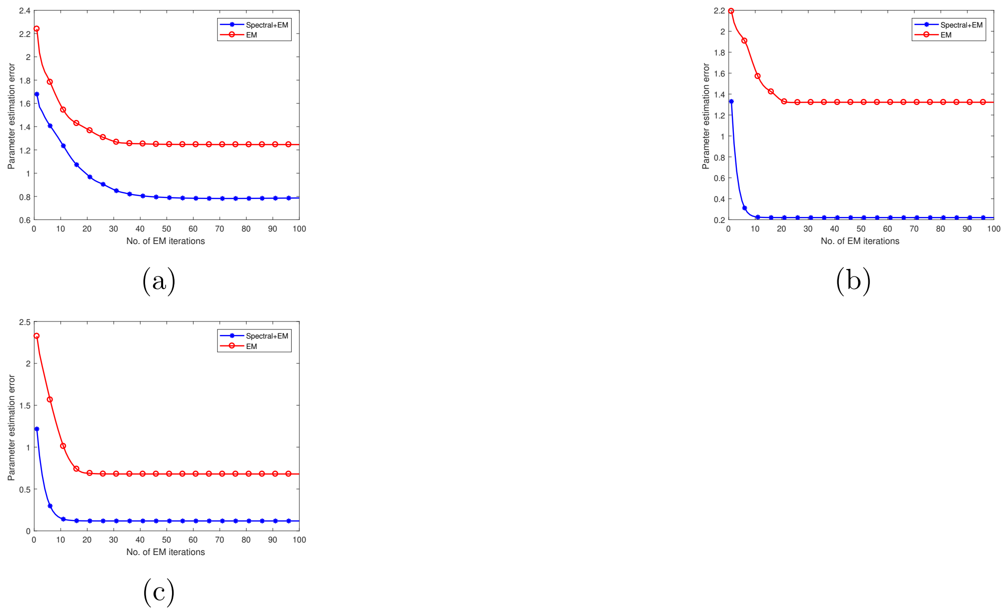


Figure A.1: Plot of parameter estimation error with varying number of samples(n): (a) $n = 1000$ (b) $n = 5000$. (c) $n = 10000$.

In Figure A.1, we varied the number of samples our data set and fixed the other set of parameters to $k = 3, d = 5, \sigma = 0.5$.

In Figure A.2 we repeated our experiments for the choice of $n = 10000, d = 5, k = 3$ for two different popular choices of non-linearities: sigmoid and ReLU. The same conclusion as in the linear setting holds in this case too with our algorithm outperforming the EM consistently.

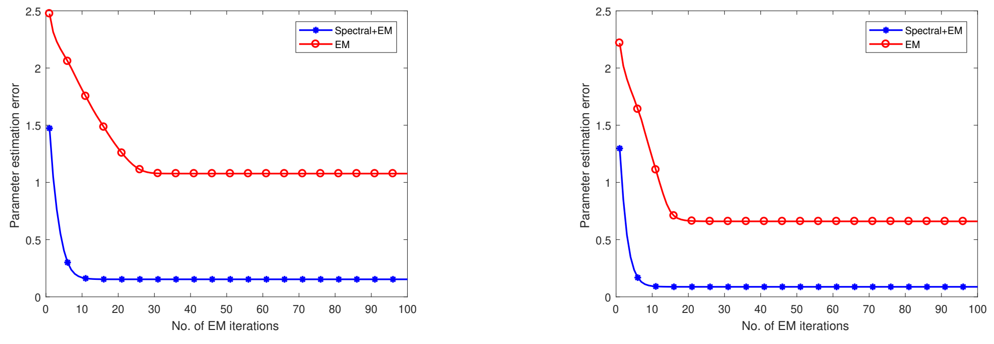


Figure A.2: Parameter estimation error for the sigmoid and ReLU nonlinearities respectively.

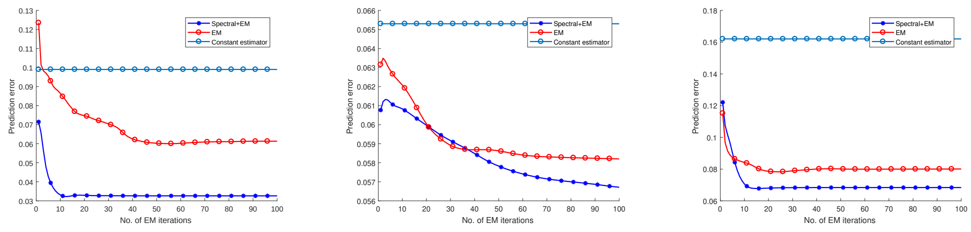


Figure A.3: Prediction error for the concrete, stock portfolio and the airfoil data sets respectively.

A.12.2 Real data

For real data experiments, we choose the three standard regression data sets from the UCI Machine Learning Repository: Concrete Compressive Strength Data Set, Stock portfolio performance Data Set, and Airfoil Self-Noise Data Set [74–76]. In all the three tasks, the goal is to predict the outcome or the response y for each input $\boldsymbol{x}$, which typically contains some task specific attributes. For example, in the concrete compressive strength, the task is to predict the compressive strength of the concrete given its various attributes such as the component of cement, water, age, etc. For this data, the input $\boldsymbol{x} \in \mathbb{R}^8$ corresponds to eight different attributes of the concrete and the output $y \in \mathbb{R}$ corresponds to its concrete strength. Similarly, for the stock portfolio data set the input $\boldsymbol{x} \in \mathbb{R}^6$ contains the weights of several stock-picking concepts such as weight of the Large S/P concept, weight of the Small systematic Risk concept, etc,. and the output y is the corresponding excess return. The airfoil data set is obtained from a series of aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections and the goal is predict the scaled sound pressure level (in dB) given the frequency, angle of attack, etc,. For all the tasks, we pre-processed the data by whitening the input and scaling the output to lie in $(-1, 1)$. We randomly allotted 75% of the data samples for training and the rest for testing. Our evaluation metric is the prediction error on the test set $(\boldsymbol{x}_i, y_i)_{i=1}^n$ defined as

$$\mathcal{E} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2,$$

where $\hat{y}_i$ corresponds to the predicted output response using the learned parameters. In other words,

$$\hat{y} = \sum_{i \in [k]} \frac{e^{\hat{\boldsymbol{w}}_i^\top \boldsymbol{x}}}{\sum_{j \in [k]} e^{\hat{\boldsymbol{w}}_j^\top \boldsymbol{x}}} \cdot g(\hat{\boldsymbol{a}}_i^\top \boldsymbol{x}).$$

We ran the joint-EM algorithm (with 10 different trails) on these tasks with various choices for $k \in \{2, \ldots, 10\}, \sigma \in \{0.1, 0.4, 0.8, 1\}, g \in \{\text{linear}, \text{sigmoid}, \text{ReLU}\}$ and found the best hyper-parameters to be ($k = 3, \sigma = 0.1$ and $g = \text{linear}$), ($k = 3, \sigma = 0.4, g = \text{sigmoid}$) and ($k = 3, \sigma = 0.1, g = \text{linear}$) for the three datasets respectively. For this choice of best hyper-parameters found for joint-EM, we ran our algorithm. Figure A.3 highlights the predictive performance of our algorithm as compared to that of the EM. We also plotted the variance of the test data for reference and to gauge the performance of our algorithm. In all the settings our algorithm is able to obtain a better set of parameters resulting in smaller prediction error.

Appendix B

Appendix for Chapter 3

B.1 Valid class of non-linearities

We slightly modify the class of non-linearities from [43] for our theoretical results in Chapter 3. The only key modification is that we use a fourth-order derivative based conditions, as opposed to third-order derivatives used in the above work. Following their notation, let $Z \sim \mathcal{N}(0,1)$ and $Y|Z \sim \mathcal{N}(g(Z), \sigma^2)$, where $g : \mathbb{R} \to \mathbb{R}$. For $(\alpha, \beta, \gamma, \delta) \in \mathbb{R}^4$, define

$$\mathcal{Q}_4(y) \triangleq Y^4 + \alpha Y^3 + \beta Y^2 + \gamma Y,$$

where

$$\begin{aligned}\mathcal{S}_4(Z) &\triangleq \mathbb{E}[\mathcal{Q}_4(y)|Z] \\ &= g(Z)^4 + 6g(Z)^2\sigma^2 + \sigma^4 + \alpha(g(Z)^3 + 3g(Z)\sigma^2) + \beta(g(Z)^2 + \sigma^2) + \gamma g(Z).\end{aligned}$$

Similarly, define

$$\mathcal{Q}_2(y) \triangleq Y^2 + \delta Y, \quad \mathcal{S}_2(Z) = \mathbb{E}[\mathcal{Q}_2(y)|Z] = g(Z)^2 + \delta g(Z) + \sigma^2.$$

Condition 3. $\mathbb{E}[\mathcal{S}_4'(Z)] = \mathbb{E}[\mathcal{S}_4''(Z)] = \mathbb{E}[\mathcal{S}_4'''(Z)] = 0$ and $\mathbb{E}[\mathcal{S}_4''''(Z)] \neq 0$. Or equivalently, in view of Stein's lemma [73],

$$\begin{aligned}\mathbb{E}[\mathcal{S}_4(Z)Z] = \mathbb{E}[\mathcal{S}_4(Z)(Z^2-1)] = \mathbb{E}[\mathcal{S}_4(Z)(Z^3-3Z)] = 0,\\ \mathbb{E}[\mathcal{S}_4(Z)(Z^4-6Z^2+3)] \neq 0.\end{aligned}$$

Condition 4. $\mathbb{E}[\mathcal{S}_2'(Z)] = 0$ and $\mathbb{E}[\mathcal{S}_2''(Z)] \neq 0$. Or equivalently,

$$\mathbb{E}[\mathcal{S}_2(Z)Z] = 0 \text{ and } \mathbb{E}[\mathcal{S}_2(Z)(Z^2-1)] \neq 0.$$

Definition 2. *We say that the non-linearity g is $(\alpha, \beta, \gamma, \delta)-$ valid if there exists a tuple $(\alpha, \beta, \gamma, \delta) \in \mathbb{R}^4$ such that both Condition 3 and Condition 4 are satisfied.*

While these conditions might seem restrictive at first, all the widely used non-linearities such as Id, ReLU, leaky-ReLU, sigmoid, etc. belong to this. For some of these non-linear activations, we provide the pre-computed transformations below:

Example 4. *If $g = \mathrm{Id}$, then $\mathcal{S}_3(y) = y^4 - 6y^2(1+\sigma^2)$ and $\mathcal{Q}_2(y) = y^2$.*

Example 5. *If $g = ReLU$, i.e. $g(z) = \max\{0, z\}$, we have that for any $p, q \in \mathbb{N}$,*

$$\mathbb{E}[g(Z)^p Z^q] = \int_0^\infty z^{p+q}\left(\frac{1}{\sqrt{2\pi}} e^{-z^2/2}\right) dz = \frac{1}{2}\mathbb{E}[|Z|^{p+q}]$$
$$= \frac{(p+q-1)!!}{2}\begin{cases}\sqrt{\frac{2}{\pi}} & \text{if } p+q \text{ is odd} \\ 1 & \text{if } p+q \text{ is even}\end{cases}.$$

Substituting these moments in the linear set of equations $\mathbb{E}[\mathcal{S}_4(Z)Z] = \mathbb{E}[\mathcal{S}_4(Z)(Z^2-1)] = \mathbb{E}[\mathcal{S}_4(Z)(Z^3-3Z)] = 0$, we obtain

$$\begin{bmatrix} 1.5 + 1.5\sigma^2 & \sqrt{\frac{2}{\pi}} + \sigma^2 & 0.5 \\ 3\sqrt{\frac{2}{\pi}}(1+\sigma^2/2) & 1+\sigma^2 & \frac{1}{2}\sqrt{\frac{2}{\pi}} \\ 3 & \sqrt{\frac{2}{\pi}} + \sigma^2 & 0 \end{bmatrix}\begin{bmatrix}\alpha \\ \beta \\ \gamma\end{bmatrix} = -\begin{bmatrix}\sqrt{\frac{2}{\pi}}(4+6\sigma^2)) \\ 6+6\sigma^2 \\ \sqrt{\frac{2}{\pi}}(12+6\sigma^2)\end{bmatrix}.$$

Solving for (α, β, γ) will yield $\mathcal{S}_4(Z)$. Finally, we have that $\delta = -2\sqrt{\frac{2}{\pi}}$.

B.2 Proofs of Section 3.2.1

Remark 5. To choose the parameters in Theorem 3, we follow the parameter choices from [41]. Let c be a sufficiently small universal constant (e.g. $c = 0.01$). Assume $\mu \le c/\kappa^*$, and $\lambda \ge 1/(ca^*_{\min})$. Let $\tau_0 = c\min\{\mu/(\kappa d a^*_{\max}), \lambda\}\,\sigma_{\min}(\boldsymbol{M})$. Let $\delta \le \min\left\{\frac{c\varepsilon_0}{a^*_{\max}\cdot m\sqrt{d}\kappa^{1/2}(\boldsymbol{M})}, \tau_0/2\right\}$ and $\varepsilon = \min\left\{\lambda\sigma_{\min}(\boldsymbol{M})^{1/2}, c\delta/\sqrt{\|\boldsymbol{M}\|}, c\varepsilon_0\delta\sigma_{\min}(\boldsymbol{M})\right\}$.

For any $k \times d$ matrix $\boldsymbol{A}$, let $\boldsymbol{A}^\dagger$ be its pseudo inverse such that $\boldsymbol{A}\boldsymbol{A}^\dagger = \boldsymbol{I}_{k\times k}$ and $\boldsymbol{A}^\dagger\boldsymbol{A}$ is the projection matrix to the row span of $\boldsymbol{A}$. Let $\alpha^*_i \triangleq \mathbb{E}[p^*_i(\boldsymbol{x})], a^*_i = \frac{1}{\alpha^*_i}$ and $\kappa^* = \frac{\alpha^*_{\max}}{\alpha^*_{\min}}$. Let $\boldsymbol{M} = \sum_{i\in[k]} \alpha^*_i \boldsymbol{a}^*_i(\boldsymbol{a}^*_i)^\top$, $\kappa(\boldsymbol{M}) = \frac{\|\boldsymbol{M}\|}{\sigma_{\min}(\boldsymbol{M})}$.

For the sake of clarity, we now formally state our main assumptions, adapted from [43]:

1. $\boldsymbol{x}$ follows a standard Gaussian distribution, i.e. $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_d)$.
2. $\|\boldsymbol{a}^*_i\| = 1$ for all $i \in [k]$ and $\|\boldsymbol{w}^*_i\| \le R$ for all $i \in [k-1]$.
3. The regressors $\boldsymbol{a}^*_1, \ldots, \boldsymbol{a}^*_k$ are linearly independent and the classifiers $\{\boldsymbol{w}^*_i\}_{i\in[k-1]}$ are orthogonal to the span $\mathcal{S} = \operatorname{span}\{\boldsymbol{a}^*_1, \ldots, \boldsymbol{a}^*_k\}$, and $2k-1 < d$.
4. The non-linearity $g : \mathbb{R} \to \mathbb{R}$ is $(\alpha, \beta, \gamma, \delta)$ – valid, which we define in Appendix B.1.

Note that while the first three assumptions are same as that of [43], the fourth assumption is slightly different from theirs. Under this assumptions, we first give an alternative characterization of $L_4(\cdot)$ in the following theorem which would be crucial for the proof of Theorem 3.

Theorem 9. *The function $L(\cdot)$ defined in Equation 3.2 satisfies that*

$$L_4(\boldsymbol{A}) = \sum_{m\in[k]} \mathbb{E}[p^*_m(\boldsymbol{x})] \sum_{\substack{i\neq j\\ i,j\in[k]}} \langle \boldsymbol{a}^*_m, \boldsymbol{a}_i\rangle^2 \langle \boldsymbol{a}^*_m, \boldsymbol{a}_j\rangle^2 - \mu \sum_{m,i\in[k]} \mathbb{E}[p^*_m(\boldsymbol{x})]\langle \boldsymbol{a}^*_m, \boldsymbol{a}_i\rangle^4$$
$$+\lambda \sum_{i\in[k]} \Big(\sum_{m\in[k]} \mathbb{E}[p^*_m(\boldsymbol{x})]\langle \boldsymbol{a}^*_m, \boldsymbol{a}_i\rangle^2 - 1\Big)^2 + \frac{\delta}{2}\|\boldsymbol{A}\|_F^2$$

B.2.1 Proof of Theorem 9

Proof. For the proof of Theorem 9, we use the notion of score functions defined as [39]:

$$\mathcal{S}_m(\boldsymbol{x}) \triangleq (-1)^m \frac{\nabla_{\boldsymbol{x}}^{(m)} f(\boldsymbol{x})}{f(\boldsymbol{x})}, \quad f \text{ is the pdf of } \boldsymbol{x}. \tag{B.1}$$

In this paper we focus on $m = 2, 4$. When $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_d)$, we know that $\mathcal{S}_2(\boldsymbol{x}) = \boldsymbol{x} \otimes \boldsymbol{x} - \boldsymbol{I}$ and

$$\mathcal{S}_4(\boldsymbol{x}) = \boldsymbol{x}^{\otimes 4} - \sum_{i \in [d]} \operatorname{sym}\left(\boldsymbol{x} \otimes \boldsymbol{e}_i \otimes \boldsymbol{e}_i \otimes \boldsymbol{x}\right) + \sum_{i,j} \operatorname{sym}\left(\boldsymbol{e}_i \otimes \boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_j\right).$$

The score transformations $\mathcal{S}_4(\boldsymbol{x})$ and $\mathcal{S}_2(\boldsymbol{x})$ can be viewed as multi-variate polynomials in $\boldsymbol{x}$ of degrees four and two respectively. For the output y, recall the transforms $\mathcal{Q}_4(y)$ and $\mathcal{Q}_2(y)$ defined in Section 3.2.1. The following lemma shows that one can construct a fourth-order super symmetric tensor using these special transforms.

Lemma 6 (Super symmetric tensor construction)**.** *Let* $(\boldsymbol{x}, y)$ *be generated according to Equation 2.1 and Assumptions* (1)-(4) *hold. Then*

$$\mathcal{T}_4 \triangleq \mathbb{E}[\mathcal{Q}_4(y) \cdot \mathcal{S}_4(\boldsymbol{x})] = c_{g,\sigma} \sum_{i \in [k]} \mathbb{E}[p_i^*(\boldsymbol{x})] \cdot \boldsymbol{a}_i^* \otimes \boldsymbol{a}_i^* \otimes \boldsymbol{a}_i^* \otimes \boldsymbol{a}_i^*,$$
$$\mathcal{T}_2 \triangleq \mathbb{E}[\mathcal{Q}_2(y) \cdot \mathcal{S}_2(\boldsymbol{x})] = c'_{g,\sigma} \sum_{i \in [k]} \mathbb{E}[p_i^*(\boldsymbol{x})] \cdot \boldsymbol{a}_i^* \otimes \boldsymbol{a}_i^*,$$

where $p_i^*(x) = \mathbb{P}\left(z_i = 1 | x\right)$, $c_{g,\sigma}$ *and* $c'_{g,\sigma}$ *are two non-zero constants depending on* g *and* σ.

Now the proof of the theorem immediately follows from Lemma 6. Recall from Equation 3.2 that

$$L_4(\boldsymbol{A}) \triangleq \sum_{\substack{i,j \in [k] \\ i \neq j}} \mathbb{E}[\mathcal{Q}_4(y) t_1(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{x})] - \mu \sum_{i \in [k]} \mathbb{E}[\mathcal{Q}_4(y) t_2(\boldsymbol{a}_i, \boldsymbol{x})]$$
$$+ \lambda \sum_{i \in [k]} \left(\mathbb{E}[\mathcal{Q}_2(y) t_3(\boldsymbol{a}_i, \boldsymbol{x})] - 1\right)^2 + \frac{\delta}{2} \|\boldsymbol{A}\|_F^2.$$

Fix $i, j \in [k]$. Notice that we have $t_1(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{x}) = \mathcal{S}_4(\boldsymbol{x})(\boldsymbol{a}_i, \boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{a}_j)/c_{g,\sigma}$. Hence we obtain

$$\begin{aligned}
\mathbb{E}[\mathcal{Q}_4(y) t_1(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{x})] &= \frac{1}{c_{g,\sigma}} \mathbb{E}[\mathcal{Q}_4(y) \cdot \mathcal{S}_4(\boldsymbol{x})](\boldsymbol{a}_i, \boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{a}_j) \\
&= \left(\sum_{m \in [k]} \mathbb{E}[p_m^*(\boldsymbol{x})](\boldsymbol{a}_m^*)^{\otimes 4} \right) (\boldsymbol{a}_i, \boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{a}_j) \\
&= \sum_{m \in [k]} \mathbb{E}[p_m^*(\boldsymbol{x})] \langle \boldsymbol{a}_m^*, \boldsymbol{a}_i \rangle^2 \langle \boldsymbol{a}_m^*, \boldsymbol{a}_j \rangle^2.
\end{aligned}$$

The simplification for the remaining terms is similar and follows directly from definitions of $t_2(\cdot, \boldsymbol{x})$ and $t_3(\cdot, \boldsymbol{x})$. □

B.2.2 Proof of Theorem 3

Proof. The proof is an immediate consequence of Theorem 9 and Theorem C.5 of [41]. □

B.2.3 Proof of Theorem 4

Proof. Note that our loss function $L_4(\boldsymbol{A})$ can be written as $\mathbb{E}[\ell(\boldsymbol{x}, y, \boldsymbol{A})]$ where ℓ is at most a fourth degree polynomial in $\boldsymbol{x}$, y and $\boldsymbol{A}$. Hence our finite sample guarantees directly follow from Theorem 3 and Theorem E.1 of [41]. □

B.2.4 Proof of Lemma 6

Proof. The proof of this lemma essentially follows the same arguments as that of [43, Theorem 1], where we replace $(\mathcal{S}_3(x), \mathcal{S}_2(x), \mathcal{P}_3(y), \mathcal{P}_2(y))$ with $(\mathcal{S}_4(x), \mathcal{S}_2(x), \mathcal{Q}_4(y), \mathcal{P}_2(y))$ respectively and letting $\mathcal{T}_3$ defined there with our $\mathcal{T}_4$ defined above.

□

B.3 Proofs of Section 3.2.2

For the convergence analysis of SGD on $L_{\log}$, we use techniques from [27] and [43]. In particular, we adapt [43, Lemma 3] and [43, Lemma 4] to our setting through Lemma 7 and Lemma 8, which are central to the proof of Theorem 5 and Theorem 6. We now sate our lemmas.

Lemma 7. *Under the assumptions of Theorem 5, it holds that*

$$\|G(\boldsymbol{W}, \boldsymbol{A}^*) - \boldsymbol{W}_i^*\| \leq \rho_\sigma \|\boldsymbol{W} - \boldsymbol{W}^*\|.$$

In addition, $\boldsymbol{W} = \boldsymbol{W}^*$ is a fixed point for $G(\boldsymbol{W}, \boldsymbol{A}^*)$.

Lemma 8. *Let the matrix of regressors $\boldsymbol{A}$ be such that $\max_{i\in[k]} \|\boldsymbol{A}_i^\top - (\boldsymbol{A}_i^*)^\top\|_2 = \sigma^2\varepsilon$. Then for any $\boldsymbol{W} \in \Omega$, we have that*

$$\|G(\boldsymbol{W}, \boldsymbol{A}) - G(\boldsymbol{W}, \boldsymbol{A}^*)\| \leq \kappa\varepsilon,$$

where κ is a constant depending on g, k and σ. In particular, $\kappa \leq (k-1)\frac{\sqrt{6(2+\sigma^2)}}{2}$ for $g =$linear, sigmoid and ReLU.

Lemma 9 (Deviation of finite sample gradient operator)**.** *For some universal constant c_1, let the number of samples n be such that $n \geq c_1 d\log(1/\delta)$. Then for any fixed set of regressors $\boldsymbol{A} \in \mathbb{R}^{k\times d}$, and a fixed $\boldsymbol{W} \in \Omega$, the bound*

$$\|G_n(\boldsymbol{W}, \boldsymbol{A}) - G(\boldsymbol{W}, \boldsymbol{A})\| \leq \varepsilon_G(n, \delta) \triangleq c_2\sqrt{\frac{d\log(k/\delta)}{n}}$$

holds with probability at least $1-\delta$.

B.3.1 Proof of Theorem 5

Proof. The proof directly follows from Lemma 7 and Lemma 8. □

B.3.2 Proof of Theorem 6

Proof. Let the set of regressors $\boldsymbol{A}$ be such that $\max_{i\in[k]} \|\boldsymbol{A}_i^\top - (\boldsymbol{A}_i^*)^\top\|_2 = \sigma^2\varepsilon_1$. Fix $\boldsymbol{A}$. For any iteration $t \in [T]$, from Lemma 9 we have the bound

$$\|G_{n/T}(\boldsymbol{W}_t, \boldsymbol{A}) - G(\boldsymbol{W}_t, \boldsymbol{A})\| \leq \varepsilon_G(n/T, \delta/T) \tag{B.2}$$

with probability at least $1 - \delta/T$. Using an union bound argument, Equation B.2 holds with probability at least $1 - \delta$ for all $t \in [T]$. Now we show that the following bound holds:

$$\|\boldsymbol{W}_{t+1} - \boldsymbol{W}^*\| \leq \rho_\sigma \|\boldsymbol{W}_t - \boldsymbol{W}^*\| + \kappa\varepsilon_1 + \varepsilon_G(n/T, \delta/T), \quad \forall t \in \{0, \dots, T-1\}. \tag{B.3}$$

Indeed, for any $t \in \{0, \dots, T-1\}$, we have that

$$\begin{aligned}
\|\boldsymbol{W}_{t+1} - \boldsymbol{W}^*\| &= \|G_{n/T}(\boldsymbol{W}_t, \boldsymbol{A}) - \boldsymbol{W}^*\| \\
&\leq \|G_{n/T}(\boldsymbol{W}_t, \boldsymbol{A}) - G(\boldsymbol{W}_t, \boldsymbol{A})\| + \|G(\boldsymbol{W}_t, \boldsymbol{A}) - G(\boldsymbol{W}_t, \boldsymbol{A}^*)\| \\
&\qquad + \|G(\boldsymbol{W}_t, \boldsymbol{A}^*) - \boldsymbol{W}^*\| \\
&\leq \varepsilon_G(n/T, \delta/T) + \kappa\varepsilon_1 + \rho_\sigma \|\boldsymbol{W}_t - \boldsymbol{W}^*\|,
\end{aligned}$$

where we used in Lemma 7, Lemma 8 and Lemma 9 in the last inequality to bound each of the terms. From Equation B.2, we obtain that

$$\begin{aligned}
\|\boldsymbol{W}_t - \boldsymbol{W}^*\| &\leq \rho_\sigma \|\boldsymbol{W}_{t-1} - \boldsymbol{W}^*\| + \kappa\varepsilon_1 + \varepsilon_G(n/T, \delta/T) \\
&\leq \rho_\sigma^2 \|\boldsymbol{W}_{t-2} - \boldsymbol{W}^*\| + (1 + \rho_\sigma)\left(\kappa\varepsilon_1 + \varepsilon_G(n/T, \delta/T)\right) \\
&\leq \rho_\sigma^t \|\boldsymbol{W}_0 - \boldsymbol{W}^*\| + \left(\sum_{s=0}^{t-1} \rho_\sigma^s\right)\left(\kappa\varepsilon_1 + \varepsilon_G(n/T, \delta/T)\right) \\
&\leq \rho_\sigma^t \|\boldsymbol{W}_0 - \boldsymbol{W}^*\| + \left(\frac{1}{1 - \rho_\sigma}\right)\left(\kappa\varepsilon_1 + \varepsilon_G(n/T, \delta/T)\right).
\end{aligned}$$

□

B.3.3 Proof of Lemma 7

Proof. Recall that the loss function for the population setting, $L_{\log}(\boldsymbol{W}, \boldsymbol{A})$, is given by

$$\begin{aligned} L_{\log}(\boldsymbol{W}, \boldsymbol{A}) &= -\mathbb{E} \log \left(\sum_{i \in [k]} \frac{e^{\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle}}{\sum_{j \in [k]} e^{\langle \boldsymbol{w}_j, \boldsymbol{x} \rangle}} \cdot \mathcal{N}(y | g(\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle), \sigma^2) \right) \\ &= -\mathbb{E} \log \left(\sum_{i \in [k]} p_i(\boldsymbol{x}) N_i \right), \end{aligned}$$

where $p_i(\boldsymbol{x}) \triangleq \frac{e^{\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle}}{\sum_{j \in [k]} e^{\langle \boldsymbol{w}_j, \boldsymbol{x} \rangle}}$ and $N_i \triangleq \mathcal{N}(y | g(\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle), \sigma^2)$. Hence for any $i \in [k-1]$, we have

$$\nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}) = -\mathbb{E} \left(\frac{\nabla_{\boldsymbol{w}_i} p_i(\boldsymbol{x}) N_i + \sum_{j \neq i, j \in [k]} \nabla_{\boldsymbol{w}_i} p_j(\boldsymbol{x}) N_j}{\sum_{i \in [k]} p_i(\boldsymbol{x}) N_i} \right).$$

Moreover,

$$\nabla_{\boldsymbol{w}_i} p_j(\boldsymbol{x}) = \begin{cases} p_i(\boldsymbol{x})(1 - p_i(\boldsymbol{x}))\boldsymbol{x}, & j = i \\ -p_i(\boldsymbol{x}) p_j(\boldsymbol{x}) \boldsymbol{x}, & j \neq i \end{cases}.$$

Hence we obtain that

$$\nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}) = -\mathbb{E} \left[\frac{p_i(\boldsymbol{x}) N_i}{\sum_{i \in [k]} p_i(\boldsymbol{x}) N_i} - p_i(\boldsymbol{x}) \right]. \tag{B.4}$$

Notice that if $z \in [k]$ denotes the latent variable corresponding to which expert is chosen, we have that the posterior probability of choosing the ith expert is given by

$$\mathbb{P}(z = i | \boldsymbol{x}, y) = \frac{p_i(\boldsymbol{x}) N_i}{\sum_{i \in [k]} p_i(\boldsymbol{x}) N_i},$$

whereas,

$$\mathbb{P}(z = i | \boldsymbol{x}) = p_i(\boldsymbol{x}).$$

Hence, when $\boldsymbol{A} = \boldsymbol{A}^*$ and $\boldsymbol{W} = \boldsymbol{W}^*$, we get that

$$\begin{aligned}\nabla_{\boldsymbol{w}_i^*} L_{\log}(\boldsymbol{W}^*, \boldsymbol{A}^*) &= -\mathbb{E}[\mathbb{P}\left(z = i|\boldsymbol{x}, y\right) - \mathbb{P}\left(z = i|\boldsymbol{x}\right)] \\ &= -\mathbb{E}[\mathbb{P}\left(z = i|\boldsymbol{x}\right) + \mathbb{E}[\mathbb{P}\left(z = i|\boldsymbol{x}\right)] = 0.\end{aligned}$$

Thus $\boldsymbol{W} = \boldsymbol{W}^*$ is a fixed point for $G(\boldsymbol{W}, \boldsymbol{A}^*)$ since

$$G(\boldsymbol{W}^*, \boldsymbol{A}^*) = \Pi_\Omega(\boldsymbol{W}^* - \alpha \nabla_{\boldsymbol{W}^*} L_{\log}(\boldsymbol{W}^*, \boldsymbol{A}^*)) = \boldsymbol{W}^*.$$

Now we make the observation that the population-gradient updates $\boldsymbol{W}_{t+1} = G(\boldsymbol{W}_t, \boldsymbol{A})$ are same as the gradient-EM updates. Thus the contraction of the population-gradient operator $G(\cdot, \boldsymbol{A}^*)$ follows from the contraction property of the gradient EM algorithm [43, Lemma 3]. To see this, recall that for k-MoE, the gradient-EM algorithm involves computing the function $Q(\boldsymbol{W}|\boldsymbol{W}_t)$ for the current iterate $\boldsymbol{W}_t$ and defined as:

$$Q(\boldsymbol{W}|\boldsymbol{W}_t) = \mathbb{E}\left[\sum_{i \in [k-1]} p^{(i)}_{\boldsymbol{W}_t}(\boldsymbol{w}_i^\top \boldsymbol{x}) - \log\left(1 + \sum_{i \in [k-1]} e^{\boldsymbol{w}_i^\top \boldsymbol{x}}\right)\right],$$

where $p^{(i)}_{\boldsymbol{W}_t} = \mathbb{P}\left(z = i|\boldsymbol{x}, y, \boldsymbol{w}_t\right)$ corresponds to the posterior probability for the i^{th} expert, given by

$$p^{(i)}_{\boldsymbol{W}_t} = \frac{p_{i,t}(\boldsymbol{x})\mathcal{N}(y|g(\boldsymbol{a}_i^\top \boldsymbol{x}), \sigma^2)}{\sum_{j\in[k]} p_{j,t}(\boldsymbol{x})\mathcal{N}(y|g(\boldsymbol{a}_j^\top \boldsymbol{x}), \sigma^2)}, \quad p_{i,t}(\boldsymbol{x}) = \frac{e^{(\boldsymbol{w}_t)_i^\top \boldsymbol{x}}}{1 + \sum_{j\in[k-1]} e^{(\boldsymbol{w}_t)_j^\top \boldsymbol{x}}}.$$

Then the next iterate of the gradient-EM algorithm is given by $\boldsymbol{W}_{t+1} = \Pi_\Omega(\boldsymbol{W}_t + \alpha \nabla_{\boldsymbol{W}} Q(\boldsymbol{W}|\boldsymbol{W}_t)_{\boldsymbol{W}=\boldsymbol{W}_t})$. We have that

$$\begin{aligned}\nabla_{\boldsymbol{w}_i} Q(\boldsymbol{W}|\boldsymbol{W}_t)|_{\boldsymbol{W}=\boldsymbol{W}_t} &= \mathbb{E}\left[\left(p^{(i)}_{\boldsymbol{W}_t} - \frac{e^{(\boldsymbol{w}_t)_i^\top \boldsymbol{x}}}{1 + \sum_{j\in[k-1]} e^{(\boldsymbol{w}_t)_j^\top \boldsymbol{x}}}\right)\boldsymbol{x}\right] \\ &= -\nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}_t, \boldsymbol{A}).\end{aligned}$$

Hence if we use the same step size α, our population-gradient iterates on the log-likelihood are same as that of the gradient-EM iterates. This finishes the proof. □

B.3.4 Proof of Lemma 8

Proof. Fix any $\boldsymbol{W} \in \Omega$ and let $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1^\top \\ \dots \\ \boldsymbol{a}_k^\top \end{bmatrix} \in \mathbb{R}^{k \times d}$ be such that $\max_{i \in [k]} \|\boldsymbol{a}_i - \boldsymbol{a}_i^*\|_2 = \sigma^2 \varepsilon_1$ for some $\varepsilon_1 > 0$. Let

$$\boldsymbol{W}' = G(\boldsymbol{W}, \boldsymbol{A}), \quad (\boldsymbol{W}')^* = G(\boldsymbol{W}, \boldsymbol{A}^*).$$

Denoting the i^{th} row of $\boldsymbol{W}' \in \mathbb{R}^{(k-1)\times d}$ by $\boldsymbol{w}_i'$ and that of $(\boldsymbol{W}')^*$ by $(\boldsymbol{w}_i')^*$ for any $i \in [k-1]$, we have that

$$\begin{aligned} \|\boldsymbol{w}_i' - (\boldsymbol{w}_i')^*\|_2 &= \|\Pi_\Omega(\boldsymbol{w}_i - \alpha \nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A})) - \Pi_\Omega(\boldsymbol{w}_i - \alpha \nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}^*))\|_2 \\ &\leq \alpha \|\nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}) - \nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}^*)\|_2. \end{aligned}$$

Thus it suffices to bound $\|\nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}) - \nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}^*)\|_2$. From Equation B.4, we have that

$$\begin{aligned} \nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}) &= -\mathbb{E}\left[\left(\frac{p_i(\boldsymbol{x}) N_i}{\sum_{i \in [k]} p_i(\boldsymbol{x}) N_i} - p_i(\boldsymbol{x})\right) \boldsymbol{x}\right], \\ \nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}^*) &= -\mathbb{E}\left[\left(\frac{p_i(\boldsymbol{x}) N_i^*}{\sum_{i \in [k]} p_i(\boldsymbol{x}) N_i^*} - p_i(\boldsymbol{x})\right) \boldsymbol{x}\right], \end{aligned}$$

where,

$$\begin{aligned} p_i(\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_i^\top \boldsymbol{x}}}{1 + \sum_{k \in [k-1]} e^{\boldsymbol{w}_j^\top \boldsymbol{x}}}, \quad N_i &\triangleq \mathcal{N}(y | g(\boldsymbol{a}_i^\top \boldsymbol{x}), \sigma^2), \\ N_i^* &= \mathcal{N}(y | g((\boldsymbol{a}_i^*)^\top \boldsymbol{x}), \sigma^2). \end{aligned}$$

Thus we have

$$\|\nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}) - \nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W}, \boldsymbol{A}^*)\|_2 = \|\mathbb{E}[(p^{(i)}(\boldsymbol{A}, \boldsymbol{W}) - p^{(i)}(\boldsymbol{A}^*, \boldsymbol{W}))\boldsymbol{x}]\|_2, \tag{B.5}$$

where $p^{(i)}(\boldsymbol{A}, \boldsymbol{W}) \triangleq \frac{p_i(\boldsymbol{x}) N_i}{\sum_{i \in [k]} p_i(\boldsymbol{x}) N_i}$ denotes the posterior probability of choosing the i^{th} expert. Now we observe that Equation B.5 reduces to the setting of [43, Lemma 4] and hence the conclusion follows.

□

B.3.5 Proof of Lemma 9

Proof. We first prove the lemma for $k = 2$. For 2-MoE, we have that the posterior probability is given by

$$p_{\boldsymbol{w}}(\boldsymbol{x}, y) = \frac{f(\boldsymbol{w}^\top \boldsymbol{x})N_1}{f(\boldsymbol{w}^\top \boldsymbol{x})N_1 + (1 - f(\boldsymbol{w}^\top \boldsymbol{x}))N_2},$$

where $f(\cdot) = \frac{1}{1+e^{-(\cdot)}}$, $N_1 = \mathcal{N}(y|g(\boldsymbol{a}_1^\top \boldsymbol{x}), \sigma^2)$ and $N_2 = \mathcal{N}(y|g(\boldsymbol{a}_2^\top \boldsymbol{x}), \sigma^2)$ for fixed $\boldsymbol{a}_1, \boldsymbol{a}_2 \in \mathbb{R}^d$. Then we have that

$$\nabla_{\boldsymbol{w}} L_{\log}(\boldsymbol{w}, \boldsymbol{A}) = -\mathbb{E}[(p_{\boldsymbol{w}}(\boldsymbol{x}, y) - f(\boldsymbol{w}^\top \boldsymbol{x})) \cdot \boldsymbol{x}].$$

Hence

$$\begin{aligned} G(\boldsymbol{w}, \boldsymbol{A}) &= \Pi_\Omega(\boldsymbol{w} + \alpha \mathbb{E}[(p_{\boldsymbol{w}}(\boldsymbol{x}, y) - f(\boldsymbol{w}^\top \boldsymbol{x})) \cdot \boldsymbol{x}]), \\ G_n(\boldsymbol{w}, \boldsymbol{A}) &= \Pi_\Omega(\boldsymbol{w} + \frac{\alpha}{n} \sum_{i \in [n]} (p_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i) - f(\boldsymbol{w}^\top \boldsymbol{x}_i)) \cdot \boldsymbol{x}_i). \end{aligned}$$

Since $0 < \alpha < 1$, we have that

$$\begin{aligned} &\|G(\boldsymbol{w}, \boldsymbol{A}) - G_n(\boldsymbol{w}, \boldsymbol{A})\|_2 \\ &\leq \|\mathbb{E}[(p_{\boldsymbol{w}}(\boldsymbol{x}, y) - f(\boldsymbol{w}^\top \boldsymbol{x}))\boldsymbol{x}] - \frac{1}{n} \sum_{i \in [n]} (p_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i) - f(\boldsymbol{w}^\top \boldsymbol{x}_i))\boldsymbol{x}_i\|_2 \\ &\leq \underbrace{\|\mathbb{E}[p_{\boldsymbol{w}}(\boldsymbol{x}, y)\boldsymbol{x}] - \sum_{i \in [n]} \frac{p_{\boldsymbol{w}}(\boldsymbol{x}_i, y_i)\boldsymbol{x}_i}{n}\|_2}_{T_1} \\ &\quad + \underbrace{\|\mathbb{E}[f(\boldsymbol{w}^\top \boldsymbol{x})\boldsymbol{x}] - \sum_{i \in [n]} \frac{f(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i}{n}\|_2}_{T_2}. \end{aligned}$$

We now bound T_1 and T_2.

Bounding T_2: We prove that the random variable $\sum_{i\in[n]} \frac{f(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i}{n} - \mathbb{E}[f(\boldsymbol{w}^\top \boldsymbol{x})\boldsymbol{x}]$ is sub-gaussian with parameter $L/\sqrt{n}$ for some constant $L > 1$ and thus its squared norm is sub-exponential. We then bound T_2 using standard sub-exponential concentration bounds. Towards the same, we first show that the random variable $f(\boldsymbol{w}^\top \boldsymbol{x})\boldsymbol{x} - \mathbb{E}[f(\boldsymbol{w}^\top \boldsymbol{x})\boldsymbol{x}]$ is sub-gaussian with parameter L. Or equivalently, that $f(\boldsymbol{w}^\top \boldsymbol{x})\langle \boldsymbol{x}, \boldsymbol{u}\rangle - \mathbb{E}[f(\boldsymbol{w}^\top \boldsymbol{x})\langle \boldsymbol{x}, \boldsymbol{u}\rangle]$ is sub-gaussian for all $\boldsymbol{u} \in \mathbb{S}^d$.

Without loss of generality, assume that $\boldsymbol{w} \neq 0$. First let $u = \vec{\boldsymbol{w}} \triangleq \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$. Thus $Z \triangleq \langle \vec{\boldsymbol{w}}, \boldsymbol{x}\rangle \sim \mathcal{N}(0,1)$. We have

$$g(Z) \triangleq f(\boldsymbol{w}^\top \boldsymbol{x})\langle \boldsymbol{x}, \vec{\boldsymbol{w}}\rangle - \mathbb{E}[f(\boldsymbol{w}^\top \boldsymbol{x})\langle \boldsymbol{x}, \vec{\boldsymbol{w}}\rangle] = f(\|\boldsymbol{w}\|Z)Z - \mathbb{E}[f(\|\boldsymbol{w}\|Z)Z].$$

It follows that $g(\cdot)$ is Lipschitz since

$$\begin{aligned} |g'(z)| = |f'(\|\boldsymbol{w}\|z)\|\boldsymbol{w}\|z + f(\|\boldsymbol{w}\|z))| &\leq \sup_{t\in\mathbb{R}} |f'(t)t| + 1 \\ &= \sup_{t>0} \frac{te^t}{(1+e^t)^2} + 1 \triangleq L. \end{aligned}$$

From the Talagaran concentration of Gaussian measure for Lipschitz functions [77], it follows that $g(Z)$ is sub-gaussian with parameter L. Now consider any $\boldsymbol{u} \in \mathbb{S}^d$ such that $\boldsymbol{u} \perp \boldsymbol{w}$. Then we have that $Y \triangleq \langle \boldsymbol{u}, \boldsymbol{x}\rangle \sim \mathcal{N}(0,1)$ and $Z \triangleq \langle \vec{\boldsymbol{w}}, \boldsymbol{x}\rangle \sim \mathcal{N}(0,1)$ are independent. Thus,

$$g(Y,Z) \triangleq f(\boldsymbol{w}^\top \boldsymbol{x})\langle \boldsymbol{u}, \boldsymbol{x}\rangle - \mathbb{E}[f(\boldsymbol{w}^\top \boldsymbol{x})\langle \boldsymbol{u}, \boldsymbol{x}\rangle] = f(\|\boldsymbol{w}\|Z)Y - \mathbb{E}[f(\|\boldsymbol{w}\|Z)Y]$$

is sub-gaussian with parameter 1 since $f \in [0,1]$ and Y, Z are independent standard gaussians. Since any $\boldsymbol{u} \in \mathbb{S}^d$ can be written as

$$u = P_{\boldsymbol{w}}(u) + P_{\boldsymbol{w}^\perp}(u),$$

where P_S denotes the projection operator onto the sub-space S, we have that $f(\boldsymbol{w}^\top \boldsymbol{x})\langle \boldsymbol{x}, \boldsymbol{u}\rangle - \mathbb{E}[f(\boldsymbol{w}^\top \boldsymbol{x})\langle \boldsymbol{x}, \boldsymbol{u}\rangle]$ is sub-gaussian with parameter L for all $\boldsymbol{u} \in \mathbb{S}^d$. Thus it follows that $\sum_{i\in[n]} \frac{f(\boldsymbol{w}^\top \boldsymbol{x}_i)\boldsymbol{x}_i}{n} - \mathbb{E}[f(\boldsymbol{w}^\top \boldsymbol{x})\boldsymbol{x}]$ is zero-mean and sub-gaussian with parameter $L/\sqrt{n}$ which further implies that

$$T_2 \leq c_2 L \sqrt{\frac{d \log(1/\delta)}{n}},$$

with probability at least $1-\delta/2$.

Bounding T_1: Let $Z \triangleq \|\sum_{i\in[n]} \frac{p_{\boldsymbol{w}}(\boldsymbol{x}_i,y_i)\boldsymbol{x}_i}{n} - \mathbb{E}[p_{\boldsymbol{w}}(\boldsymbol{x},y)\boldsymbol{x}]\|_2 = \sup_{\boldsymbol{u}\in\mathbb{S}^d} Z(\boldsymbol{u})$, where

$$Z(\boldsymbol{u}) \triangleq \sum_{i\in[n]} \frac{p_{\boldsymbol{w}}(\boldsymbol{x}_i,y_i)\langle \boldsymbol{x}_i,\boldsymbol{u}\rangle}{n} - \mathbb{E}[p_{\boldsymbol{w}}(\boldsymbol{x},y)\langle \boldsymbol{x},\boldsymbol{u}\rangle].$$

Let $\{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_M\}$ be a 1/2-cover of the unit sphere $\mathbb{S}^d$. Hence for any $\boldsymbol{v}\in\mathbb{S}^d$, there exists a $j\in[M]$ such that $\|\boldsymbol{v}-\boldsymbol{u}_j\|_2 \le 1/2$. Thus,

$$Z(\boldsymbol{v}) \le Z(\boldsymbol{u}_j) + |Z(\boldsymbol{v}) - Z(\boldsymbol{u}_j)| \le Z\|\boldsymbol{v}-\boldsymbol{u}_j\|_2 \le Z(\boldsymbol{u}_j) + Z/2,$$

where we used the fact that $|Z(\boldsymbol{u}) - Z(\boldsymbol{v})| \le Z\|\boldsymbol{u}-\boldsymbol{v}\|_2$ for any $\boldsymbol{u},\boldsymbol{v}\in\mathbb{S}^d$. Now taking supremum over all $\boldsymbol{v}\in\mathbb{S}^d$ yields that $Z \le 2\max_{j\in[M]} Z(\boldsymbol{u}_j)$. Now we bound $Z(\boldsymbol{u})$ for a fixed $\boldsymbol{u}\in\mathbb{S}^d$. By symmetrization trick [78], we have

$$\mathbb{P}\left(Z(\boldsymbol{u}) \ge t\right) \le 2\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^n \varepsilon_i p_{\boldsymbol{w}}(\boldsymbol{x}_i,y_i)\langle \boldsymbol{x}_i,\boldsymbol{u}\rangle \ge t/2\right),$$

where $\varepsilon_1,\ldots,\varepsilon_n$ are i.i.d. Rademacher variables. Define the event $E \triangleq \{\frac{1}{n}\sum_{i\in[n]}\langle \boldsymbol{x}_i,\boldsymbol{u}\rangle^2 \le 2\}$. Since $\langle \boldsymbol{x}_i,\boldsymbol{u}\rangle \sim \mathcal{N}(0,1)$, standard tail bounds imply that $\mathbb{P}\left(E^c\right) \le e^{-n/32}$. Thus we have that

$$\mathbb{P}\left(Z(\boldsymbol{u}) \ge t\right) \le 2\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^n \varepsilon_i p_{\boldsymbol{w}}(\boldsymbol{x}_i,y_i)\langle \boldsymbol{x}_i,\boldsymbol{u}\rangle \ge t/2|E\right) + 2e^{-n/32}.$$

Considering the first term, for any $\lambda>0$, we have

$$\mathbb{E}[\exp\left(\frac{\lambda}{n}\sum_{i=1}^n \varepsilon_i p_{\boldsymbol{w}}(\boldsymbol{x}_i,y_i)\langle \boldsymbol{x}_i,\boldsymbol{u}\rangle\right)|E] \le \mathbb{E}[\exp\left(\frac{2\lambda}{n}\sum_{i=1}^n \varepsilon_i\langle \boldsymbol{x}_i,\boldsymbol{u}\rangle\right)|E],$$

where we used the Ledoux-Talagrand contraction for Rademacher process [77], since $|p_{\boldsymbol{w}}(\boldsymbol{x}_i,y_i)| \le 1$ for all $(\boldsymbol{x}_i,y_i)$. The sub-gaussianity of Rademacher sequence $\{\varepsilon_i\}$ implies that

$$\mathbb{E}[\exp\left(\frac{2\lambda}{n}\sum_{i=1}^n \varepsilon_i\langle \boldsymbol{x}_i,\boldsymbol{u}\rangle\right)|E] \le \mathbb{E}[\exp\left(\frac{2\lambda^2}{n^2}\sum_{i=1}^n \langle \boldsymbol{x}_i,\boldsymbol{u}\rangle^2\right)|E] \le \exp(\frac{4\lambda^2}{n}),$$

using the definition of the event E. Thus the above bound on the moment generating function implies the following tail bound:

$$\mathbb{P}\left(\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i p_{\boldsymbol{w}}(\boldsymbol{x}_i,y_i)\langle \boldsymbol{x}_i,\boldsymbol{u}\rangle \geq t/2|E\right) \leq \exp\left(-\frac{nt^2}{256}\right).$$

Combining all the bounds together, we obtain that

$$\mathbb{P}\left(Z(\boldsymbol{u})\geq t\right) \leq 2e^{-nt^2/256}+2e^{-n/32}.$$

Since $M \leq 2^d$, using the union bound we obtain that

$$\mathbb{P}\left(Z\geq t\right) \leq 2^d(2e^{-nt^2/1024}+2e^{-n/32}).$$

Since $n \geq c_1 d\log(1/\delta)$, we have that $T_1 = Z \leq c\sqrt{\frac{d\log(1/\delta)}{n}}$ with probability at least $1-\delta/2$. Combining these bounds on T_1 and T_2 yields the final bound on $\varepsilon_G(n,\delta)$.

Now consider any $k \geq 2$. From Equation B.4, defining $N_i \triangleq \mathcal{N}(y|g(\boldsymbol{a}_i^\top \boldsymbol{x}),\sigma^2)$ and $p_i(\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_i^\top \boldsymbol{x}}}{1+\sum_{j\in[k-1]}e^{\boldsymbol{w}_j^\top \boldsymbol{x}}}$, we have that

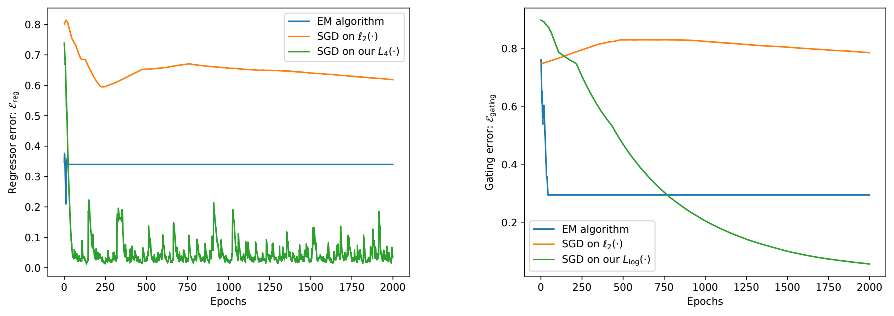
$$\nabla_{\boldsymbol{w}_i} L_{\log}(\boldsymbol{W},\boldsymbol{A}) = -\mathbb{E}\left(\frac{p_i(\boldsymbol{x})N_i}{\sum_{i\in[k]}p_i(\boldsymbol{x})N_i}-p_i(\boldsymbol{x})\right)\boldsymbol{x}.$$

Similarly,

$$\nabla_{\boldsymbol{w}_i} L^{(n)}_{\log}(\boldsymbol{W},\boldsymbol{A}) = -\sum_{j=1}^{n}\frac{1}{n}\left(\frac{p_i(\boldsymbol{x}_j)N_i}{\sum_{i\in[k]}p_i(\boldsymbol{x}_j)N_i}-p_i(\boldsymbol{x}_j)\right)\boldsymbol{x}_j.$$

Since $\|G_n(\boldsymbol{W},\boldsymbol{A})-G(\boldsymbol{W},\boldsymbol{A})\| = \max_{i\in[k-1]}\|G_n(\boldsymbol{W},\boldsymbol{A})_i-G(\boldsymbol{W},\boldsymbol{A})_i\|_2$, with out loss of generality, we let $i=1$. The proof for the other cases is similar. Thus we have

$$\begin{aligned}\|G_n(\boldsymbol{W},\boldsymbol{A})_1-G(\boldsymbol{W},\boldsymbol{A})_1\|_2 &\leq \|\nabla_{\boldsymbol{w}_1}L_{\log}(\boldsymbol{W},\boldsymbol{A})-\nabla_{\boldsymbol{w}_1}L^{(n)}_{\log}(\boldsymbol{W},\boldsymbol{A})\|_2\\ &\leq \|\sum_{i=1}^{n}\frac{p^{(1)}(\boldsymbol{x}_i,y_i)\boldsymbol{x}_i}{n}-\mathbb{E}[p^{(1)}(\boldsymbol{x},y)\boldsymbol{x}]\|_2\\ &\quad+\|\sum_{i=1}^{n}\frac{p_1(\boldsymbol{x})\boldsymbol{x}}{n}-\mathbb{E}[p_1(\boldsymbol{x})\boldsymbol{x}]\|_2,\end{aligned}$$



(a) Regressor error (b) Gating error

Figure B.1: Comparison of SGD on our losses $(L_4, L_{\log})$ vs. ℓ_2 and the EM algorithm.

where $p^{(1)}(\boldsymbol{x}, y) \triangleq \frac{p_1(\boldsymbol{x})N_1}{\sum_{i\in[k]} p_i(\boldsymbol{x})N_i}$. Since $|p^{(1}(\boldsymbol{x}, y)| \leq 1$ and $|p_1(\boldsymbol{x})| \leq 1$, we can use the same argument as in the bounding of T_1 proof for 2-MoE above to get the parametric bound. This finishes the proof. □

B.4 Additional experiments

B.4.1 Reduced batch size

In Figure B.1 we ran SGD on our loss $L_4(\cdot)$ with five different runs with a batch size of 128 and a learning rate of 0.001 for $d = 10$ and $k = 3$. We can see that our algorithm still converges to zero but with more variance because of noisy gradient estimation and also lesser number of samples than the required sample complexity.

Appendix C

Appendix for Chapter 4

C.1 Polar$(64, 7)$ code

Recall from Section 4.5 that the Plotkin tree for a Polar code is obtained by freezing a set of leaves in a complete binary tree. These frozen leaves are chosen according to the reliabilities, or equivalently, error probabilities of their corresponding bit channels. In other words, we first approximate the error probabilities of all the n-bit channels and pick the k-smallest of them using the procedure from [79]. These k active set of leaves correspond to the transmitted message bits, whereas the remaining $n - k$ frozen leaves always transmit zero.

Here we focus on a specific Polar code: Polar$(64, 7)$, with code dimension $k = 7$ and blocklength $n = 64$. For Polar$(64, 7)$, we obtain these active set of leaves to be $\mathcal{A} = \{48, 56, 60, 61, 62, 63, 64\}$, and the frozen set to be $\mathcal{A}^c = \{1, 2, \cdots 64\} \backslash \mathcal{A}$. Using these set of indices and simplifying the redundant branches, we obtain the Plotkin tree for Polar$(64, 7)$ to be Figure C.1. We observe that this Polar Plotkin tree shares some similarities with that of a RM$(6, 1)$ code (with same $k = 7$ and $n = 64$) with key differences at the topmost and bottom most leaves.

Capitalizing on the encoding tree structure of Polar$(64, 7)$, we build a corresponding KO encoder g_θ which inherits this tree skeleton. In other words, we generalize the Plotkin mapping blocks at the internal nodes of tree, except for the root node, and replace them with a corresponding neural network g_i. Figure C.1 depicts the Plotkin tree of our KO encoder. The KO decoder f_ϕ is designed similarly. Training of the (encoder, decoder) pair (g_θ, f_ϕ) is similar to that of the KO$(8, 2)$ training which we detail in §4.4.

Figure C.2 shows the BLER performance of the Polar$(64, 7)$ code and its competing KO code for the AWGN channel. Similar to the BER performance analyzed in Figure 4.10, the KO code is able to significantly improve the BLER performance. For example, we achieve a gain of around 0.5 dB when KO encoder is combined with the MAP decoding. Additionally, the close performance of the KO decoder to that of the MAP decoder confirms its optimality.

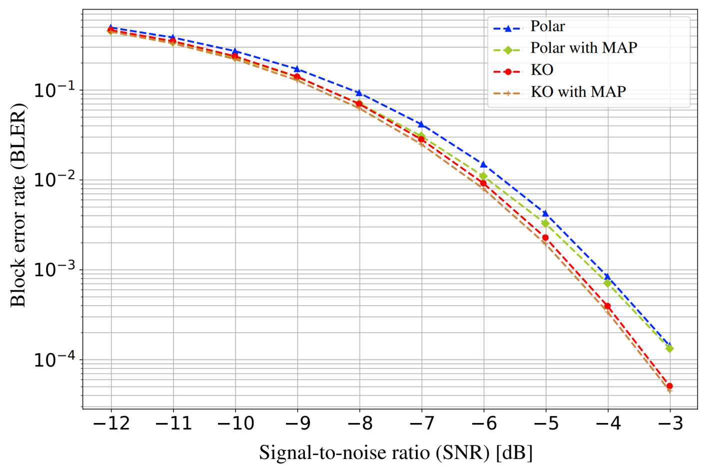


Figure C.2: KO code achieves a significant gain over the Polar$(64, 7)$ code in BLER when trained on AWGN channel. KO decoder also matches the optimal MAP decoder.

C.2 Gains with list decoding

Successive cancellation decoding can be significantly improved by list decoding. List decoding allows one to gracefully trade off computational complexity and reliability by maintaining a list (of a fixed size) of candidate codewords during the decoding process. The following figure demonstrates that KO(8,2) code with list decoding enjoys a significant gain over the non-list counterpart. This promising result opens several interesting directions which are current focuses of active research.

Polar codes with list decoding achieves the state-of-the-art performances [54]. It is a promising direction to design large block-lengths KO codes (based on the skeleton of Polar codes) that can improve upon the state-of-the-art list-decoded Polar codes. One direction is to train KO codes as we propose and include list decoding after the training. A more ambitious direction is to include list decoding in the training, potentially further improving the performance by discovering an encoder tailored for list decoding.

Unlike Polar codes, RM codes have an extra structure of an algebraic symmetry; a RM codebook is invariant under certain permutations. This can be exploited in list decoding as shown in [1], to get a further gain over what is shown in Figure C.3. However, when a KO code is trained based on a RM skeleton, this symmetry is lost. A question of interest is whether one can discover nonlinear codes with such symmetry.

C.3 Discussion

C.3.1 On modulation and practicality of KO codes

We note that our real-valued KO codewords are *entirely practical* in wireless communication; the peak energy of a symbol is only 22.48% larger than the average in KO codes. The impact on the power amplifier is not any different from that of a more traditional modulation (like 16-QAM). Training KO code is a form of *jointly* designing the coding and modulation steps; this approach has a long history in wireless communication (e.g., Trellis coded modulation) but the performance gains have been restricted by the human ingenuity in constructing the heuristics.

C.3.2 Comparison with LDPC and BCH codes

We expect good performance for BCH at the short blocklengths considered in the paper though with high-complexity (polynomial time) decoders such as ordered statistics decoder (OSD). On the other hand, there does not exist good LDPC codes at the k and n regimes of this paper; thus, we do not expect good performance for LDPC at these regimes.

C.4 Plotkin construction

[50] proposed this scheme in order to combine two codes of smaller code lengths and construct a larger code with the following properties. It is relatively easy to construct a code with either a high rate but a small distance (such as sending the raw information bits directly) or a large distance but a low rate (such as repeating each bit multiple times). Plotkin construction combines such two codes of rates $\rho_{\boldsymbol{u}} > \rho_{\boldsymbol{v}}$ and distances $d_{\boldsymbol{u}} < d_{\boldsymbol{v}}$, to design a larger block length code satisfying rate $\rho = (\rho_{\boldsymbol{u}} + \rho_{\boldsymbol{v}})/2$ and distance $\min\{2d_{\boldsymbol{u}}, d_{\boldsymbol{v}}\}$. This significantly improves upon a simple time-sharing of those codes, which achieves the same rate but distance only $\min\{d_{\boldsymbol{u}}, d_{\boldsymbol{v}}\}$.

Note: Following the standard convention, we fix the leaves in the Plotkin tree of a first order RM$(m, 1)$ code to be zeroth order RM codes and the full-rate RM$(1, 1)$ code. On the other hand, a second order RM$(m, 2)$ code contains the first order RM codes and the full-rate RM$(2, 2)$ as its leaves.

C.5 KO$(8, 2)$: Architecture and training

As highlighted in §4.4, our KO codes improve upon RM codes significantly on a variety of benchmarks. We present the architectures of the KO$(8, 2)$ encoder, the KO$(8, 2)$ decoder, and their joint training methodology that are crucial for this superior performance.

C.5.1 KO$(8,2)$ encoder

Architecture. KO$(8,2)$ encoder inherits the same Plotkin tree structure as that of the second order RM$(8,2)$ code and thus RM codes of first order and the second order RM$(2,2)$ code constitute the leaves of this tree, as highlighted in Figure C.4b. On the other hand, a critical distinguishing component of our KO$(8,2)$ encoder is a set of encoding neural networks $g_\theta = \{g_1, \ldots, g_6\}$ that strictly generalize the Plotkin mapping. In other words, we associate a neural network $g_i \in g_\theta$ to each internal node i of this tree. If $\boldsymbol{v}$ and $\boldsymbol{u}$ denote the codewords arriving from left and right branches at this node, we combine them non-linearly via the operation $(\boldsymbol{u}, \boldsymbol{v}) \mapsto g_i(\boldsymbol{u}, \boldsymbol{v})$.

We carefully parametrize each encoding neural network g_i so that they generalize the classical Plotkin map Plotkin$(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}, \boldsymbol{u} \oplus \boldsymbol{v})$. In particular, we represent them as $g_i(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}, \widetilde{g}_i(\boldsymbol{u}, \boldsymbol{v}) + \boldsymbol{u} \oplus \boldsymbol{v})$, where $\widetilde{g}_i : \mathbb{R}^2 \to \mathbb{R}$ is a neural network of input dimension 2 and output size 1. Here $\widetilde{g}_i$ is applied coordinate-wise on its inputs $\boldsymbol{u}$ and $\boldsymbol{v}$. This clever parametrization can also be viewed as a skip connection on top of the Plotkin map. Similar skip-like ideas have been successfully used in the literature though in a different context of learning decoders [10]. On the other hand, we exploit these ideas for both encoders and decoders which further contribute to significant gains over RM codes.

Encoding. From an encoding perspective, recall that the KO$(8,2)$ code has code dimension $k = 37$ and block length $n = 256$. Suppose we wish to transmit a set of 37 message bits denoted as $\boldsymbol{m} = (\boldsymbol{m}_{(2,2)}, \boldsymbol{m}_{(2,1)}, \ldots, \boldsymbol{m}_{(7,1)})$ through our KO$(8,2)$ encoder. We first encode the block of four message bits $\boldsymbol{m}_{(2,2)}$ into a RM$(2,2)$ codeword $\boldsymbol{c}_{(2,2)}$ using its corresponding encoder at the bottom most leaf of the Plotkin tree. Similarly we encode the next three message bits $\boldsymbol{m}_{(2,1)}$ into an RM$(2,1)$ codeword $\boldsymbol{c}_{(2,1)}$. We combine these codewords using the neural network g_6 at their parent node which yields the codeword $\boldsymbol{c}_{(3,2)} = g_6(\boldsymbol{c}_{(2,2)}, \boldsymbol{c}_{(2,1)}) \in \mathbb{R}^8$. The codeword $\boldsymbol{c}_{(3,2)}$ is similarly combined with its corresponding left codeword and this procedure is thus recursively carried out till we reach the top most node of the tree which outputs the codeword $\boldsymbol{c}_{(8,2)} \in \mathbb{R}^{256}$. Finally we obtain the unit-norm KO$(8,2)$ codeword $\boldsymbol{x}$ by normalizing $\boldsymbol{c}_{(8,2)}$, i.e. $\boldsymbol{x} = \boldsymbol{c}_{(8,2)} / \|\boldsymbol{c}_{(8,2)}\|_2$.

Note that the map of encoding the message bits $\boldsymbol{m}$ into the codeword $\boldsymbol{x}$, i.e. $\boldsymbol{x} = g_\theta(\boldsymbol{m})$, is differentiable with respect to θ since all the underlying operations at each node of the Plotkin tree are differentiable.

C.5.2 KO$(8, 2)$ decoder

Architecture. Capitalizing on the recursive structure of the encoder, the KO$(8, 2)$ decoder decodes the message bits from top to bottom, similar in style to Dumer's decoding in §4.2. More specifically, at any internal node of the tree we first decode the message bits along its left branch, which we utilize to decode that of the right branch and this procedure is carried out recursively till all the bits are recovered. At the leaves, we use the Soft-MAP decoder to decode the bits.

Similar to the encoder g_θ, an important aspect of our KO$(8, 2)$ decoder is a set of decoding neural networks $f_\phi = \{f_1, f_2, \ldots, f_{11}, f_{12}\}$. For each node i in the tree, $f_{2i-1} : \mathbb{R}^2 \to \mathbb{R}$ corresponds to its left branch whereas $f_{2i} : \mathbb{R}^4 \to \mathbb{R}$ corresponds to the right branch. The pair of decoding neural networks (f_{2i-1}, f_{2i}) can be viewed as matching decoders for the corresponding encoding network g_i: While g_i encodes the left and right codewords arriving at this node, the outputs of f_{2i-1} and f_{2i} represent appropriate Euclidean feature vectors for decoding them. Further, f_{2i-1} and f_{2i} can also be viewed as a generalization of Dumer's decoding to nonlinear real codewords: f_{2i-1} generalizes the LSE function, while f_{2i} extends the operation $\oplus_{\hat{\boldsymbol{v}}}$. More precisely, we represent $f_{2i-1}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \widetilde{f}_{2i-1}(\boldsymbol{y}_1, \boldsymbol{y}_2) + \mathrm{LSE}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ whereas $f_{2i}(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_{\boldsymbol{v}}, \hat{\boldsymbol{v}}) = \widetilde{f}_{2i}(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_{\boldsymbol{v}}, \hat{\boldsymbol{v}}) + \boldsymbol{y}_1 + (-1)^{\hat{\boldsymbol{v}}}\boldsymbol{y}_2$, where $(\boldsymbol{y}_1, \boldsymbol{y}_2)$ are appropriate feature vectors from the parent node, and $\boldsymbol{y}_{\boldsymbol{v}}$ is the feature corresponding to the left-child $\boldsymbol{v}$, and $\hat{\boldsymbol{v}}$ is the decoded left-child codeword. We explain about these feature vectors in more detail below. Note that both the functions $\widetilde{f}_{2i-1}$ and $\widetilde{f}_{2i}$ are also applied coordinate-wise.

Decoding. At the decoder, suppose we receive a noisy codeword $\boldsymbol{y} \in \mathbb{R}^{256}$ at the root upon transmission of the actual codeword $\boldsymbol{x} \in \mathbb{R}^{256}$ along the channel. The first step is to obtain the LLR feature for the left RM$(7,1)$ codeword; we obtain this via the left neural network f_1, i.e. $\boldsymbol{y}_v = f_1(\boldsymbol{y}_{1:128}, \boldsymbol{y}_{129:256}) \in \mathbb{R}^{128}$. Subsequently, the Soft-MAP decoder transforms this feature into an LLR vector for the message bits, i.e. $\boldsymbol{L}_{(7,1)} = \text{Soft-MAP}(f_1(\boldsymbol{y}_{1:128}, \boldsymbol{y}_{129:256}))$. Note that the message bits $\boldsymbol{m}_{(7,1)}$ can be hard decoded directly from the sign of $\boldsymbol{L}_{(7,1)}$. Instead here we use their soft version via the sigmoid function $\sigma(\cdot)$, i.e., $\hat{\boldsymbol{m}}_{(7,1)} = \sigma(\boldsymbol{L}_{(7,1)})$. Thus we obtain the corresponding RM$(7,1)$ codeword $\hat{\boldsymbol{v}}$ by encoding the message $\hat{\boldsymbol{m}}_{(7,1)}$ via an RM$(7,1)$ encoder. The next step is to obtain the feature vector for the right child. This is done using the right decoder f_2, i.e. $\boldsymbol{y}_{\boldsymbol{u}} = f_2(\boldsymbol{y}_{1:128}, \boldsymbol{y}_{129:256}, \boldsymbol{y}_{\boldsymbol{v}}, \hat{\boldsymbol{v}})$. Utilizing this right feature $\boldsymbol{y}_{\boldsymbol{u}}$ the decoding procedure is thus recursively carried out till we compute the LLRs for all the remaining message bits $\boldsymbol{m}_{(6,1)}, \ldots, \boldsymbol{m}_{(2,2)}$ at the leaves. Finally we obtain the full LLR vector $\boldsymbol{L} = (\boldsymbol{L}_{(7,1)}, \ldots, \boldsymbol{L}_{(2,2)})$ corresponding to the message bits $\boldsymbol{m}$. A simple sigmoid transformation, $\sigma(\boldsymbol{L})$, further yields the probability of each of these message bits being zero, i.e. $\sigma(\boldsymbol{L}) = \mathbb{P}(\boldsymbol{m} = \boldsymbol{0})$.

Note that the decoding map $f_\phi : \boldsymbol{y} \mapsto \boldsymbol{L}$ is fully differentiable with respect to ϕ, which further ensures a differentiable loss for training the parameters (θ, ϕ).

C.5.3 Training

Recall that we have the following flow diagram from encoder till the decoder when we transmit the message bits $\boldsymbol{m}$: $\boldsymbol{m} \xrightarrow{g_\theta} \boldsymbol{x} \xrightarrow{Channel} \boldsymbol{y} \xrightarrow{f_\phi} \boldsymbol{L} \xrightarrow{\sigma(\cdot)} \sigma(\boldsymbol{L})$. In view of this, we define an end-to-end differentiable cross entropy loss function to train the parameters (θ, ϕ), i.e.

$$L(\theta, \phi) = \sum_j m_j \log(1 - \sigma(L_j)) + (1 - m_j) \log \sigma(L_j).$$

Finally we run Algorithm 3 on the loss $L(\theta, \phi)$ to train the parameters (θ, ϕ) via gradient descent.

C.6 Soft-MAP decoder

As discussed earlier (see also Figure C.5), Dumer's decoder for second-order RM codes RM$(m, 2)$ performs MAP decoding at the leaves while our KO decoder applies Soft-MAP decoding at the leaves. The leaves of both RM$(m, 2)$ and KO$(m, 2)$ codes are comprised of order-one RM codes and the RM$(2, 2)$ code. In this section, we first briefly state the MAP decoding rule over general binary-input memoryless channels and describe how the MAP rule can be obtained in a more efficient way, with complexity $\mathcal{O}(n \log n)$, for first-order RM codes. We then present the generic Soft-MAP decoding rule and its efficient version for first-order RM codes.

MAP decoding. Given a length-n channel LLR vector $\boldsymbol{l} \in \mathbb{R}^n$ corresponding to the transmission of a given (n, k) code, i.e. code dimension is k and block length is n, with codebook $\mathcal{C}$ over a general binary-input memoryless channel, the MAP decoder picks a codeword $\boldsymbol{c}^*$ according to the following rule [56]

$$\boldsymbol{c}^* = \operatorname*{argmax}_{\boldsymbol{c} \in \mathcal{C}} \ \langle \boldsymbol{l}, 1 - 2\boldsymbol{c} \rangle, \tag{C.1}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner-product of two vectors. Obviously, the MAP decoder needs to search over all 2^k codewords while each time computing the inner-product of two length-n vectors. Therefore, the MAP decoder has a complexity of $\mathcal{O}(n2^k)$. Thus the MAP decoder can be easily applied to decode small codebooks like an RM$(2,2)$ code, that has blocklength $n = 4$ and a dimension $k = 4$, with complexity $\mathcal{O}(1)$. On the other hand, a naive implementation of the MAP rule for RM$(m,1)$ codes, that have $2^k = 2^{m+1} = 2n$ codewords, requires $\mathcal{O}(n^2)$ complexity. However, utilizing the special structure of order-1 RM codes, one can apply the fast Hadamard transform (FHT) to implement their MAP decoding in a more efficient way, i.e., with complexity $\mathcal{O}(n \log n)$. The idea behind the FHT implementation is that the standard $n \times n$ Hadamard matrix $\mathbf{H}$ contains half of the the $2n$ codewords of an RM$(m,1)$ code (in ± 1), and the other half are just $-\mathbf{H}$. Therefore, FHT of the vector $\boldsymbol{l}$, denoted by $\boldsymbol{l}_{\text{WH}}$, lists half of the $2n$ inner-products in (C.1), and the other half are obtained the as $-\boldsymbol{l}_{\text{WH}}$. Therefore, the FHT version of the MAP decoder for first-order RM codes can be obtained as

$$\boldsymbol{c}^* = (1 - \text{sign}(\boldsymbol{l}_{\text{WH}}(i^*))\boldsymbol{h}_{i^*})/2 \quad \text{s.t.} \quad i^* = \underset{i \in [n]}{\text{argmax}} \; |\boldsymbol{l}_{\text{WH}}(i)|, \tag{C.2}$$

where $\boldsymbol{l}_{\text{WH}}(i)$ is the i-th element of the vector $\boldsymbol{l}_{\text{WH}}$, and $\boldsymbol{h}_i$ is the i-th row of the matrix $\mathbf{H}$. Given that $\boldsymbol{l}_{\text{WH}}$ can be efficiently computed with $\mathcal{O}(n \log n)$ complexity, the FHT version of the MAP decoder for the first-order RM codes, described in (C.2), has a complexity of $\mathcal{O}(n \log n)$.

Soft-MAP. Note that the MAP decoder and its FHT version involve argmax$(\cdot)$ operation which is not differentiable. In order to overcome this issue, we obtain the soft-decision version of the MAP decoder, referred to as Soft-MAP decoder, to come up with differentiable decoding at the leaves [80]. The Soft-MAP decoder obtains the soft LLRs instead of hard decoding of the codes at the leaves. Particularly, consider an AWGN channel model as $\boldsymbol{y} = \boldsymbol{s} + \boldsymbol{n}$, where $\boldsymbol{y}$ is the length-n vector of the channel output, $\boldsymbol{s} := 1 - 2\boldsymbol{c}$, $\boldsymbol{c} \in \mathcal{C}$, and $\boldsymbol{n}$ is the vector of the Gaussian noise with mean zero and variance σ^2 per element. The LLR of the i-th information bit u_i is then defined as

$$\boldsymbol{l}_{\text{inf}}(i) := \ln \left(\frac{\Pr(u_i = 0|\boldsymbol{y})}{\Pr(u_i = 1|\boldsymbol{y})} \right). \tag{C.3}$$

By applying the Bayes' rule, the assumption of $\Pr(u_i = 0) = \Pr(u_i = 1)$, the law of total probability, and the distribution of the Gaussian noise, we can write (C.3) as

$$\boldsymbol{l}_{\text{inf}}(i) = \ln \left(\frac{\sum_{\boldsymbol{s} \in \mathcal{C}_i^0} \exp\left(-||\boldsymbol{y} - \boldsymbol{s}||_2^2/\sigma^2\right)}{\sum_{\boldsymbol{s} \in \mathcal{C}_i^1} \exp\left(-||\boldsymbol{y} - \boldsymbol{s}||_2^2/\sigma^2\right)} \right). \tag{C.4}$$

We can also apply the max-log approximation to approximate (C.4) as follows.

$$\boldsymbol{l}_{\text{inf}}(i) \approx \frac{1}{\sigma^2} \min_{\boldsymbol{c} \in \mathcal{C}_i^1} ||\boldsymbol{y} - \boldsymbol{s}||_2^2 - \frac{1}{\sigma^2} \min_{\boldsymbol{c} \in \mathcal{C}_i^0} ||\boldsymbol{y} - \boldsymbol{s}||_2^2, \tag{C.5}$$

where $\mathcal{C}_i^0$ and $\mathcal{C}_i^1$ denote the subsets of codewords that have the i-th information bit u_i equal to zero and one, respectively. Finally, given that the length-n LLR vector of the cahhnel output can be obtained as $\boldsymbol{l} := 2\boldsymbol{y}/\sigma^2$ for the AWGN channels, and assuming that all the codewords $\boldsymbol{s}$'s have the same norm, we obtain a more useful version of the Soft-MAP rule for approximating the LLRs of the information bits as

$$\boldsymbol{l}_{\text{inf}}(i) \approx \max_{\boldsymbol{c} \in \mathcal{C}_i^0} \langle \boldsymbol{l}, 1 - 2\boldsymbol{c} \rangle \; - \; \max_{\boldsymbol{c} \in \mathcal{C}_i^1} \langle \boldsymbol{l}, 1 - 2\boldsymbol{c} \rangle. \tag{C.6}$$

It is worth mentioning at the end that, similar to the MAP rule, one can compute all the 2^k inner products in $\mathcal{O}(n2^k)$ time complexity, and then obtain the soft LLRs by looking at appropriate indices. As a result, the complexity of the Soft-MAP decoding for decoding RM$(m, 1)$ and RM$(2, 2)$ codes is $\mathcal{O}(n^2)$ and $\mathcal{O}(1)$ respectively. However, one can apply an approach similar to (C.2) to obtain a more efficient version of the Soft-MAP decoder, with complexity $\mathcal{O}(n \log n)$, for decoding RM$(m, 1)$ codes.

C.7 Experimental details

We provide our code at https://github.com/deepcomm/KOcodes.

Algorithm 3 Training algorithm for KO(8,2)

```
Input: number of epochs $T$, number of encoder training steps $T_{\text{enc}}$, number of decoder training steps $T_{\text{dec}}$, encoder training SNR $\text{SNR}_{\text{enc}}$, decoder training SNR $\text{SNR}_{\text{dec}}$, learning rate for encoder $\text{lr}_{\text{enc}}$, learning rate for decoder $\text{lr}_{\text{dec}}$
Initialize $(\theta, \phi)$
for $T$ steps do
    for $T_{\text{dec}}$ steps do
        Generate a minibatch of random message bits $\boldsymbol{m}$
        Simulate AWGN channel with $\text{SNR}_{\text{dec}}$
        Fix $\theta$, update $\phi$ by minimizing $L(\theta, \phi)$ using Adam with $\text{lr}_{\text{dec}}$
    end for
    for $T_{\text{enc}}$ steps do
        Generate a minibatch of random message bits $\boldsymbol{m}$
        Simulate AWGN channel with $\text{SNR}_{\text{enc}}$
        Fix $\phi$, update $\theta$ by minimizing $L(\theta, \phi)$ using Adam with $\text{lr}_{\text{enc}}$
    end for
end for
Output: $(\theta, \phi)$
```

C.7.1 Hyper-parameter choices for KO(8,2)

We choose batch size $B = 50000$, encoder training SNR $\text{SNR}_{\text{enc}} = -3dB$, decoder trainint SNR $\text{SNR}_{\text{dec}} = -5dB$, number of epochs $T = 2000$, number of encoder training steps $T_{\text{enc}} = 50$, number of decoder training steps $T_{\text{dec}} = 500$. For Adam optimizer, we choose learning rate for encoder $\text{lr}_{\text{enc}} = 10^{-5}$ and for decoder $\text{lr}_{\text{dec}} = 10^{-4}$.

C.7.2 Neural network architecture of KO(8,2)

Initialization

We design our (encoder, decoder) neural networks to generalize and build upon the classical (Plotkin map, Dumer's decoder). In particular, as discussed in Section C.5.1, we parameterize the KO encoder g_θ, as $g_i(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}, \widetilde{g}_i(\boldsymbol{u}, \boldsymbol{v}) + \boldsymbol{u} \oplus \boldsymbol{v})$, where $\widetilde{g} : \mathbb{R}^2 \to \mathbb{R}$ is a fully connected neural network, which we delineate in Section C.7.2. Similarly, for KO decoder, we parametrize it as $f_{2i-1}(\boldsymbol{y}_1, \boldsymbol{y}_2) = \widetilde{f}_{2i-1}(\boldsymbol{y}_1, \boldsymbol{y}_2) + \mathrm{LSE}(\boldsymbol{y}_1, \boldsymbol{y}_2)$ and $f_{2i}(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_{\boldsymbol{v}}, \hat{\boldsymbol{v}}) = \widetilde{f}_{2i}(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_{\boldsymbol{v}}, \hat{\boldsymbol{v}}) + \boldsymbol{y}_1 + (-1)^{\hat{\boldsymbol{v}}}\boldsymbol{y}_2$, where $\widetilde{f}_{2i-1} : \mathbb{R}^2 \to \mathbb{R}$ and $\widetilde{f}_{2i} : \mathbb{R}^4 \to \mathbb{R}$ are also fully connected neural networks whose architectures are described in Section C.7.2 and Section C.7.2. If $\widetilde{f} \approx 0$ and $\widetilde{g} \approx 0$, we are able to thus recover the standard RM$(8,2)$ encoder and its corresponding Dumer decoder. By initializing all the weight parameters (θ, ϕ) sampling from $\mathcal{N}(0, 0.02^2)$, we are able to approximately recover the performance RM$(8,2)$ at the beginning of the training which acts as a good initialization for our algorithm.

Architecture of $\widetilde{g}_i$

- Dense(units=2×32)
- SeLU()
- Dense(units=32×32)
- SeLU()
- Dense(units=32×32)
- SeLU()
- Dense(units=32×1)

Architecture of $\widetilde{f}_{2i}$

- Dense(units=4×32)
- SeLU()
- Dense(units=32×32)
- SeLU()

- Dense(units=32×32)
- SeLU()
- Dense(units=32×1)

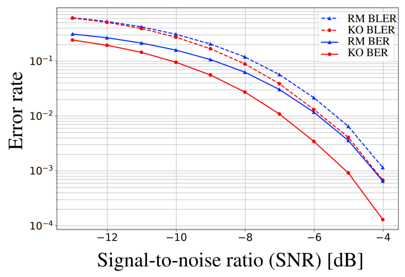
Architecture of $\widetilde{f}_{2i-1}$

- Dense(units=2×32)
- SeLU()
- Dense(units=32×32)
- SeLU()
- Dense(units=32×32)
- SeLU()
- Dense(units=32×1)

C.8 Results for Order-1 codes

Here we focus on first order KO$(m, 1)$ codes, and in particular KO$(6, 1)$ code that has code dimension k =7 and blocklength $n = 64$. The training of the (encoder, decoder) pair (g_θ, f_ϕ) for KO$(6, 1)$is almost identical to that of the second order RM$(8, 2)$ described in §4.3. The only difference is that we now use the Plotkin tree structure of the corresponding RM$(6, 1)$ code. In addition, we also train our neural encoder g_θ together with the differentiable MAP decoder, i.e. the Soft-MAP, to compare its performance to that of the RM codes. Figure C.6 illustrates these results.

The left panel of Figure C.6 highlights that KO$(6, 1)$ obtains significant gain over RM$(6, 1)$ code (with Dumer decoder) when both the neural encoder and decoder are trained jointly. On the other hand, in the right panel, we notice that we match the performance of that of the RM$(6, 1)$ code (with the MAP decoder) when we just train the encoder g_θ (with the MAP decoder). In other words, under the optimal MAP decoding, KO$(6, 1)$ and RM$(6, 1)$ codes behave the same. Note that the only caveat for KO$(6, 1)$ in the second setting is that its MAP decoding complexity is $O(n^2)$ while that of the RM is $O(n \log n)$.



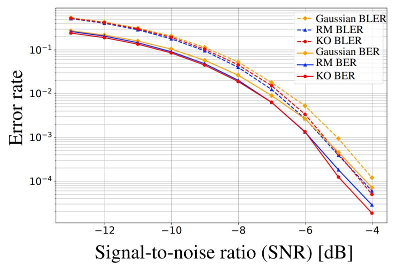
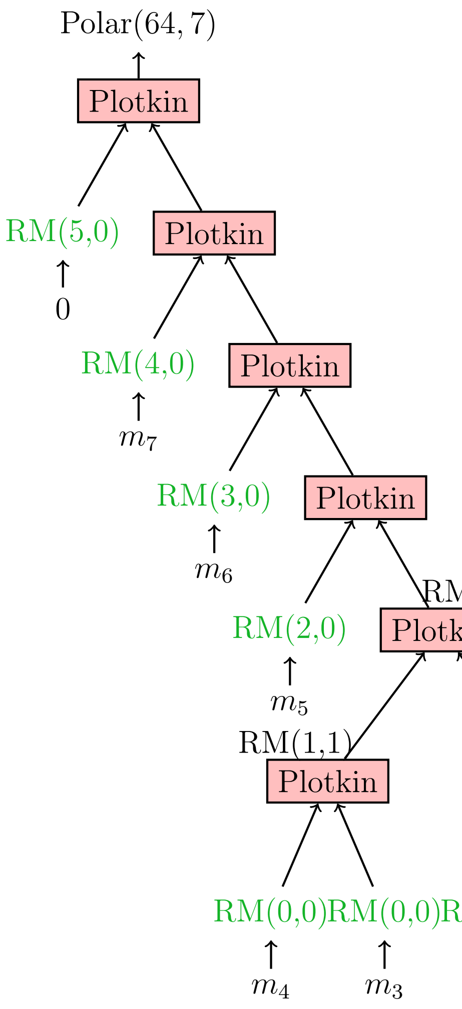
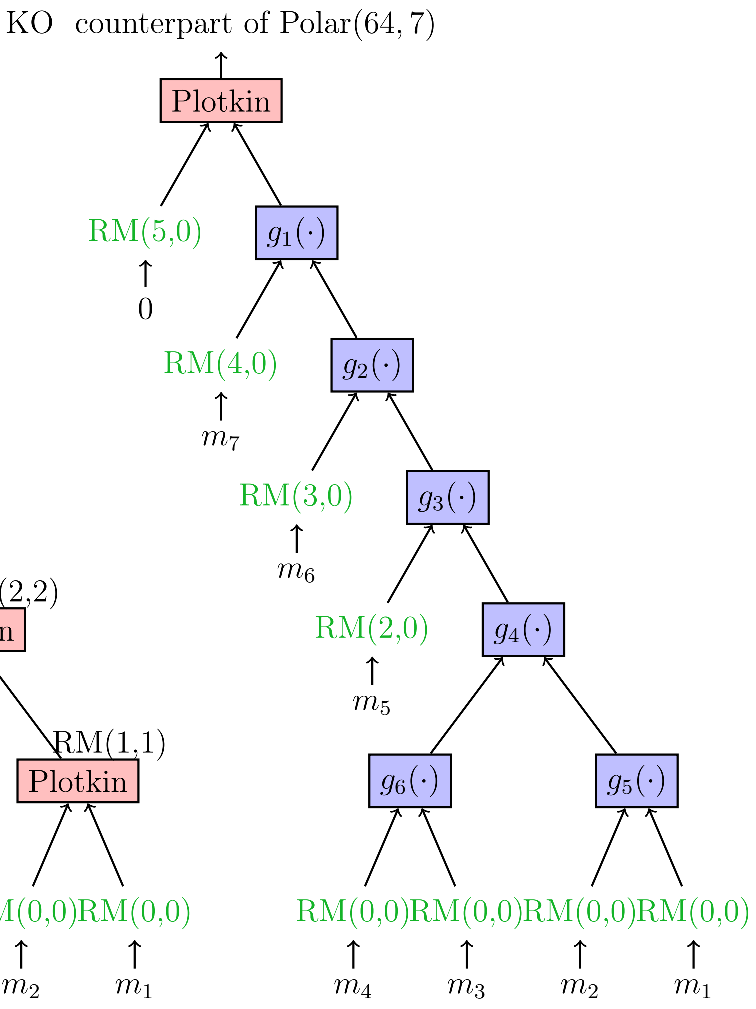


Figure C.6: KO$(6,1)$ code. **Left**: KO$(6,1)$ code achieves significant gain over RM$(6,1)$ code (with Dumer) when trained on AWGN channel. **Right**: Under the optimal MAP decoding, KO$(6,1)$ and RM$(6,1)$ codes achieve the same performance. Error rates for a random Gaussian codebook are also plotted as a baseline.



(a) Polar(64, 7) encoder



(b) Our KO encoder counterpart

Figure C.1: Plotkin trees for the Polar$(64, 7)$ encoder and our neural KO encoder counterpart. Both codes have dimension $k = 7$ and blocklength $n = 64$.

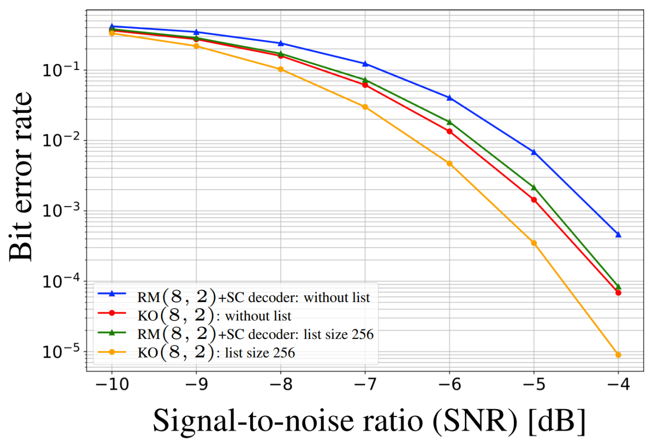
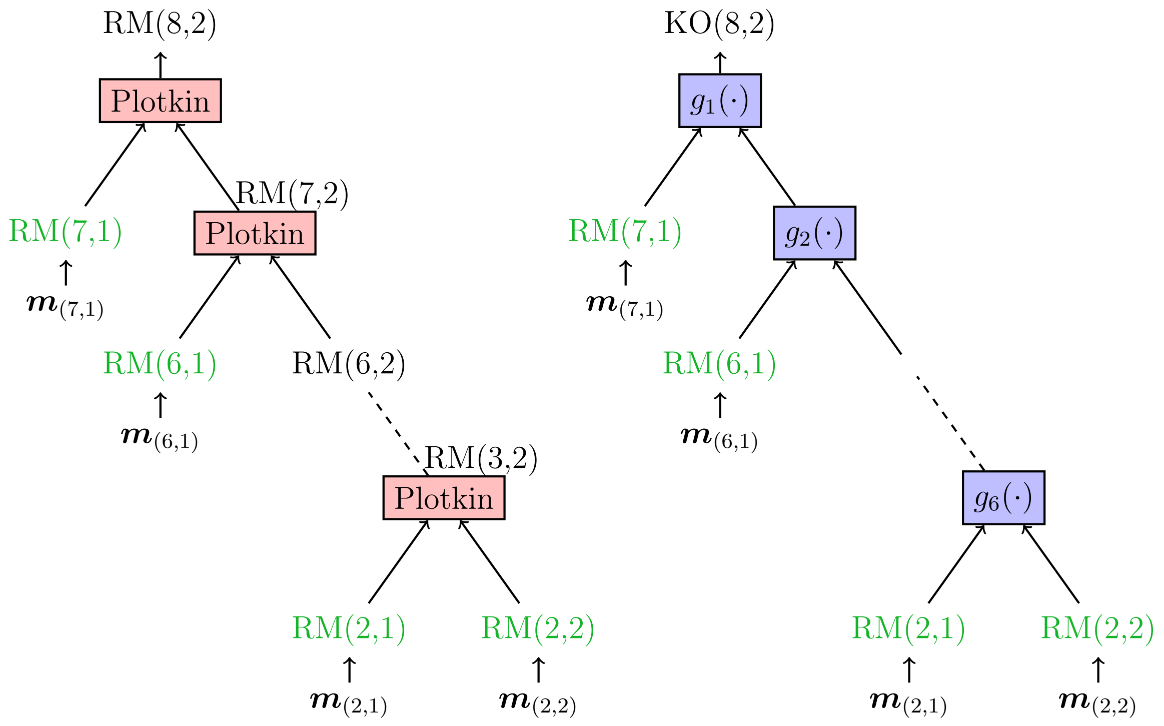
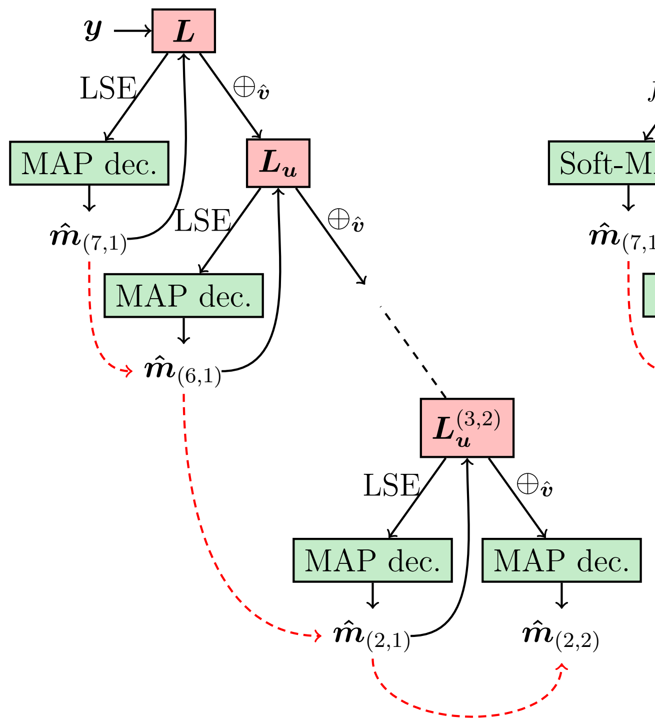


Figure C.3: The same KO(8,2) encoder and decoder as those used in Figure 4.4 achieve a significant gain (without any retraining or fine-tuning) when list decoding is used together with the KO decoder. The magnitude of the gain is comparable to the gain achieved by the same list decoding technique on the successive cancellation decoder of the RM(8,2) code. We used the list decoding from [1] but without the permutation technique.



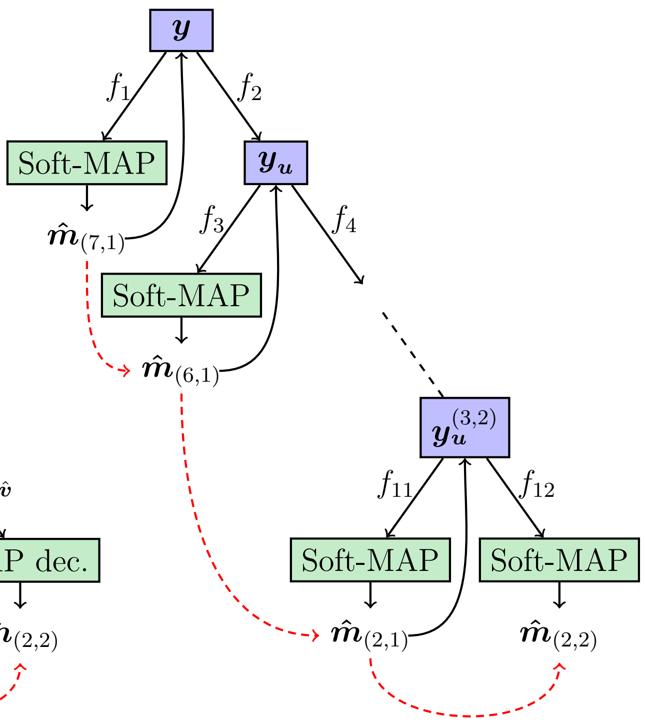
(a) RM(8,2) encoder (b) KO(8,2) encoder

Figure C.4: Plotkin trees for RM(8,2) and KO(8,2) encoders. Leaves are highlighted in green. Both codes have dimension $k = 37$ and blocklength $n = 256$.



$\boldsymbol{y}$
$\boldsymbol{L}$
LSE
$\oplus_{\hat{\boldsymbol{v}}}$
MAP dec.
$\boldsymbol{L_u}$
$\hat{\boldsymbol{m}}_{(7,1)}$
MAP dec.
$\hat{\boldsymbol{m}}_{(6,1)}$
$\boldsymbol{L_u}^{(3,2)}$
MAP dec.
MAP dec.
$\hat{\boldsymbol{m}}_{(2,1)}$
$\hat{\boldsymbol{m}}_{(2,2)}$

(a) RM$(8,2)$ decoder



$\boldsymbol{y}$
f_1
f_2
Soft-MAP
$\boldsymbol{y_u}$
f_3
f_4
$\hat{\boldsymbol{m}}_{(7,1)}$
Soft-MAP
$\hat{\boldsymbol{m}}_{(6,1)}$
$\boldsymbol{y_u}^{(3,2)}$
f_{11}
f_{12}
Soft-MAP
Soft-MAP
$\hat{\boldsymbol{m}}_{(2,1)}$
$\hat{\boldsymbol{m}}_{(2,2)}$

(b) KO$(8,2)$ decoder

Figure C.5: Plotkin trees for the RM$(8,2)$ and KO$(8,2)$ decoders. Red arrows indicate the bit decoding order.