Inference in Generalized Linear Models with Applications

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree
Doctor of Philosophy in the Graduate School of The Ohio State
University

By

Evan Byrne, B.S., M.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2019

Dissertation Committee:

Dr. Philip Schniter, Advisor

Dr. Lee C. Potter

Dr. Kiryung Lee

Abstract

Inference involving the generalized linear model is an important problem in signal processing and machine learning. In the first part of this dissertation, we consider two problems involving the generalized linear model. Specifically, we consider sparse multinomial logistic regression (SMLR) and sketched clustering, which in the context of machine learning are forms of supervised and unsupervised learning, respectively. Conventional approaches to these problems fit the parameters of the model to the data by minimizing some regularized loss function between the model and data, which typically is performed by an iterative gradient-based algorithm. While these methods generally work, they may suffer from various issues such as slow convergence or getting stuck in a sub-optimal solution. Slow convergence is particularly detrimental when applied to modern datasets, which may contain upwards of millions of sample points. We take an alternate inference approach based on approximate message passing, rather than optimization. In particular, we apply the hybrid generalized approximate message passing (HyGAMP) algorithm to both of these problems in order to learn the underlying parameters of interest. The HyGAMP algorithm approximates the sum-product or min-sum loopy belief propagation algorithms, which approximate minimum mean squared error (MMSE) or maximum a posteriori (MAP) estimation, respectively, of the unknown parameters of interest. We first apply the MMSE and

MAP forms of the HyGAMP algorithm to the SMLR problem. Next, we apply a simplified form of HyGAMP (SHyGAMP) to SMLR, where we show through numerical experiments that our approach meets or exceeds the performance of state-of-the-art SMLR algorithms with respect to classification accuracy and algorithm training time (i.e., computational efficiency). We then apply the MMSE-SHyGAMP algorithm to the sketched clustering problem, where we also show through numerical experiments that our approach exceeds the performance of other state-of-the-art sketched clustering algorithms with respect to clustering accuracy and computational efficiency. We also show our approach has better clustering accuracy and better computational efficiency than the widely used K-means++ algorithm in some regimes.

Finally, we study the problem of adaptive detection from quantized measurements. We focus on the case of strong, but low-rank interference, which is motivated by wireless communications applications for the military, where the receiver is experiencing strong jamming from a small number of sources in a time-invariant channel. In this scenario, the receiver requires many antennas to effectively null out the interference, but this comes at the cost of increased hardware complexity and increased volume of data. Using highly quantized measurements is one method of reducing the complexity of the hardware and the volume of data, but it is unknown how this method affects detection performance. We first investigate the effect of quantized measurements on existing unquantized detection algorithms. We observe that unquantized detection algorithms applied to quantized measurements lack the ability to null arbitrarily large interference, despite being able to null arbitrarily large interference when applied to unquantized measurements. We then derive a generalized likelihood ratio test for the quantized measurement model, which again gives rise to a generalized bilinear model.

Via simulation, we empirically observe the quantized algorithm only offers a fraction of a decibel improvement in equivalent SNR relative to unquantized algorithms. We then evaluate alternative techniques to address the performance loss due to quantized measurements, including a novel analog pre-whitening using digitally controlled phase-shifters. In simulation, we observe that the new technique shows up to 8 dB improvement in equivalent SNR.

Acknowledgments

Many people deserve thanks for assisting me in the process of completing this dissertation. First and foremost I must thank my advisor Phil Schniter for contributing a significant amount of time and effort towards mentoring me and for guiding my research with helpful insights and advise. I also want to thank many ECE faculty, particularly Lee Potter, for serving on my numerous committees and for teaching exciting and useful courses. I am also very grateful towards Adam Margetts, who served as a dedicated mentor and collaborator for much of my dissertation, and towards MIT Lincoln Laboratory for financially supporting a large portion of my research.

Other people in the department helped as well. Jeri McMichael and Tricia Toothman were always friendly to talk to, interested in my progress, and provided valuable help in scheduling and other areas. The other IPS students, including Jeremy Vila, Justin Ziniel, Mark Borgerding, Mike Riedl, Adam Rich, You Han, Tarek Abdal-Rahmen, Subrata Sarkar, Ted Reehorst, and Antoine Chatalic provided a welcoming community, support and collaboration.

Finally, I am very thankful for my family and friends, who supported and encouraged me the entire way.

Vita

December 2012	B.S. Electrical and Computer Engineering, The Ohio State University
August 2015	M.S. Electrical and Computer Engineering, The Ohio State University
2013-present	Graduate Research Assistant, The Ohio State University

Publications

E. Byrne and P. Schniter, "Sparse Multinomial Logistic Regression via Approximate Message Passing," *IEEE Transactions on Signal Processing*, vol. 64, no. 21, pp. 5485-5498, Nov. 2016.

S. Rangan, A. K. Fletcher, V. K. Goyal, E. Byrne, and P. Schniter, "Hybrid Approximate Message Passing," *IEEE Transactions on Signal Processing*, vol. 65, no. 17, pp. 4577-4592, Sep. 2017.

E. Byrne, R. Gribonval, and P. Schniter, "Sketched Clustering via Hybrid Approximate Message Passing," *Proc. Asilomar Conf. on Signals, Systems, and Computers* (Pacific Grove, CA), Nov. 2017.

P. Schniter and E. Byrne, "Adaptive Detection of Structured Signals in Low-Rank Interference," *IEEE Transactions on Signal Processing, accepted.*

E. Byrne, A. Chatalic, R. Gribonval, and P. Schniter, "Sketched Clustering via Hybrid Approximate Message Passing," *in review.*

Fields of Study

Major Field: Electrical and Computer Engineering

Studies in:

Machine Learning
Signal Processing

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

In this dissertation we consider a wide variety of inference problems that involve the generalized linear model (GLM). We begin with several motivating examples that stem from the prominence of "Big Data", where the datasets may be very large and the challenge is to efficiently process the data in a manner that makes useful insights.

Our first example is the feature selection problem in the field of genomics. In this problem one is given gene-expression data from several patients, each patient with a specific (known) disease, and tasked with determining which genes predict the presence of each disease. The challenging aspect of this problem is that the number of features, which in this problem are the potentially predictive genes, is very large (typically tens of thousands of genes), and, due to the cost of obtaining the data, is much larger than the number of gene-expression samples per disease (typically dozens).

Our second application is the classification problem; here we consider document classification as an example. In this problem the goal is to use the relative frequency of the various keywords to predict the subject area of future documents. In order to design a prediction rule, one is given a training dataset where each sample corresponds to a document in a specific, known, subject area, and the features of the sample are the relative frequency of various keywords. Two aspects of this problem that make it challenging are: 1) the number of different subject areas may be large (more than 10), and 2) designing a decision rule from the dataset may be computationally challenging due to its sheer size. For example, a dataset in this domain may contain hundreds of thousands of samples and each sample contains tens of thousands of features.

Another application is the clustering problem, which is similar to the classification problem, except here the datasets are "unlabeled" and the goal is to partition the samples into their appropriate classes. For example, there may be a large dataset containing images of handwritten digits, where the class corresponds to the digit (0-9), but prior to designing a classifier with this dataset the samples must first be labeled. Labeling by hand is not a feasible option due to the large quantity of samples, and more importantly, in many applications a human may not be able to correctly determine the class. Therefore, an algorithm that can accurately partition the dataset into clusters is desired.

A final application is the detection problem in array processing. Here, one may sample incoming radio signals across both space (using multiple antennas) and time (using multiple temporal snapshots) with the goal of determining the presence or absence of a specific signal while in the presence of corrupting noise and interference. Two practical uses for this problem are in radar and in wireless communications. In radar one emits a known signal and makes estimates of the scene based on properties of the reflected signal, while in communications one user emits a known synchronization signal to another user in order for a connection to be fully established. In these examples, if many antennas (dozens) and time snapshots (thousands) are used, the entire block of data may be quite large and computationally challenging to process.

In addition, practical effects such as quantization, which distort the measured signal, may have a large effect on performance and should be considered in the design.

There are many approaches to tackling the aforementioned problems, some of which involve inference with the GLM. In the sequel, we introduce the GLM and commonly applied maximum likelihood (ML) and maximum a posteriori (MAP) inference framework.

1.1 Introduction to Generalized Linear Models

1.1.1 The Standard Linear Model

Prior to describing the GLM, we first present the standard linear model, given by

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}, \tag{1.1}$$

where $\boldsymbol{y} \in \mathbb{R}^M$ is a vector of measurements, $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ is a known linear operator, and $\boldsymbol{w}$ is independent and identically distributed (iid) Gaussian noise. Our objective is to infer $\boldsymbol{x}$. There are various approaches to doing so. Perhaps the most fundamental is the maximum likelihood (ML) estimation framework, which makes no prior assumptions on $\boldsymbol{x}$ and solves

$$\begin{aligned} \widehat{\boldsymbol{x}}_{\mathsf{ML}} &= \arg\max_{\boldsymbol{x}} \log p(\boldsymbol{y}|\boldsymbol{x}) && (1.2)\\ &= \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 && (1.3)\\ &= \boldsymbol{A}^{+}\boldsymbol{y}, && (1.4) \end{aligned}$$

where $\boldsymbol{A}^{+}$ is the pseudoinverse of $\boldsymbol{A}$ and (1.3) follows from (1.2) due to the iid Gaussian assumption on $\boldsymbol{w}$.

The solution given by (1.4) is not always desirable. For example, if $M < N$, (1.4) is one of infinitely many solutions to Problem (1.2), or it may not be near the true

$\boldsymbol{x}$. However, this problem may be circumvented by maximum a posteriori (MAP) estimation, where one specifies a prior $p(\boldsymbol{x})$ that imparts a desired structure on the estimate $\widehat{\boldsymbol{x}}$. For example, in Compressed Sensing (CS), one hypothesizes that $\boldsymbol{x}$ is sparse, i.e., it contains mostly zeros, say $S < N$ non-zero elements. If $S < M$, it may be possible to find an accurate estimate $\widehat{\boldsymbol{x}}$. Therefore, one applies the Laplacian prior $p(\boldsymbol{x}) \propto \exp(-\lambda\|\boldsymbol{x}\|_1)$, which promotes learning a sparse $\widehat{\boldsymbol{x}}$. This results in the LASSO problem, given by

$$\widehat{\boldsymbol{x}}_{\mathsf{LASSO}} = \arg\min_{\boldsymbol{x}} \Big\{ \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1 \Big\}, \tag{1.5}$$

where λ controls the strength of this regularization; the larger λ, the more sparse $\widehat{\boldsymbol{x}}_{\mathsf{LASSO}}$ will be.

1.1.2 The Generalized Linear Model

In the standard linear model, the measurements $\boldsymbol{y}$ are Gaussian-noise corrupted versions of the "noiseless" measurements $\boldsymbol{z} \triangleq \boldsymbol{A}\boldsymbol{x}$. The GLM instead models $p(\boldsymbol{y}|\boldsymbol{z})$ as an arbitrary likelihood that need not correspond to additive white Gaussian noise. The objective remains to learn $\boldsymbol{x}$, possibly under prior assumptions, while using the arbitrary likelihood $p(\boldsymbol{y}|\boldsymbol{z})$.

A popular example of the GLM is logistic regression. Logistic regression is one approach to binary linear classification and feature selection. In binary linear classification, one is given M feature vectors with corresponding binary class labels $\Big\{\boldsymbol{a}_m \in \mathbb{R}^N, y_m \in \{-1, 1\}\Big\}_{m=1}^{M}$ with the goal of designing a weight vector $\boldsymbol{x}$ that accurately classifies a test feature vector $\boldsymbol{a}_0$, where classification is performed via $\widehat{y}_0 = \operatorname{sgn} \boldsymbol{a}_0^\mathsf{T}\boldsymbol{x}$. For example, if one considers the genomics problem described in the first part of this chapter, y_m may indicate one of two possible disease types, while elements in $\boldsymbol{a}_m$ represent the expression level of individual genes.

In logistic regression, we model

$$p(y_m|z_m = \boldsymbol{a}_m^\mathsf{T}\boldsymbol{x}) = \frac{1}{1+\exp(-y_m z_m)}, \tag{1.6}$$

which takes values between 0 and 1 and models that probability that $\boldsymbol{a}_m$ belongs to class y_m with the decision boundary defined by $\boldsymbol{x}$.

The ML approach to logistic regression is to solve

$$\begin{aligned} \widehat{\boldsymbol{x}}_{\mathsf{ML}} &= \arg\max_{\boldsymbol{x}} \log p(\boldsymbol{y}|\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}) && (1.7)\\ &= \arg\max_{\boldsymbol{x}} \sum_{m=1}^{M} \log \frac{1}{1+\exp(-y_m \boldsymbol{a}_m^\mathsf{T}\boldsymbol{x})}, && (1.8) \end{aligned}$$

where $\boldsymbol{a}_m^\mathsf{T}$ form the rows of $\boldsymbol{A}$. However, similar to the linear case, issues may arise with $\widehat{\boldsymbol{x}}_{\mathsf{ML}}$. For example, if the data is linearly separable, then $\widehat{\boldsymbol{x}}_{\mathsf{ML}}$ will be infinite valued.

MAP estimation offers a resolution. For example, a Gaussian prior on $\boldsymbol{x}$ (which is equivalent to quadratic regularization $\|\boldsymbol{x}\|_2^2$) will prevent the estimate from being infinite valued, even in the linearly separable case. Or, similar to the CS problem, if we can correctly hypothesize that only $S < M$ features (elements of $\boldsymbol{a}_m$) are useful for classification, accurate classification may be achieved by learning an S-sparse $\widehat{\boldsymbol{x}}$, which may be achieved by incorporating a $\lambda\|\boldsymbol{x}\|_1$ regularization, in which case we have

$$\widehat{\boldsymbol{x}}_{\mathsf{MAP}} = \arg\max_{\boldsymbol{x}} \Big\{ \log p(\boldsymbol{y}|\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}) - \lambda\|\boldsymbol{x}\|_1 \Big\}. \tag{1.9}$$

Referring back to the genomics problem, feature selection can be performed by looking at the support of sparse $\widehat{\boldsymbol{x}}_{\mathsf{MAP}}$.

GLMs with Additional Structure

So far, we have considered problems where y_m depends only on a scalar z_m. However, there exist GLMs with additional structure. One example is where $\boldsymbol{X} \in \mathbb{R}^{N \times K}$ and y_m depends on $\boldsymbol{z}_m \triangleq \boldsymbol{X}^\mathsf{T} \boldsymbol{a}_m \in \mathbb{R}^K$. An example is multinomial logistic regression, which is the multi-class extension of logistic regression. In multinomial logistic regression we still have $\boldsymbol{a}_m \in \mathbb{R}^N$, but now $y_m \in \{1, ..., K\}$, indicating to which of K classes $\boldsymbol{a}_m$ belongs. The probability that $\boldsymbol{a}_m$ belongs to class $y_m \in \{1, ..., K\}$, given decision boundaries parameterized by $\boldsymbol{X}$, is now modeled by the "softmax" equation

$$p(y_m | \boldsymbol{z}_m = \boldsymbol{X}^\mathsf{T} \boldsymbol{a}_m) = \frac{\exp\Big([\boldsymbol{z}_m]_{y_m}\Big)}{\sum_{k=1}^K \exp\Big([\boldsymbol{z}_m]_k\Big)} \tag{1.10}$$

and classification of $\boldsymbol{a}_0$ is performed via $\widehat{y}_0 = \arg\max_k [\boldsymbol{z}_0]_k$. Problems involving inference in these types of GLMs are a primary focus of this dissertation.

1.1.3 The Generalized Bilinear Model

The generalized bilinear model is similar to the GLM, except that now $\boldsymbol{A}$ and $\boldsymbol{X}$ must both be estimated. One example is binary principal components analysis (PCA) (which is discussed more in Chapter 5), where we have

$$\boldsymbol{Y} = \operatorname{sgn}(\boldsymbol{A}\boldsymbol{X} + \boldsymbol{W}) \tag{1.11}$$

and wish to infer $\boldsymbol{A}$ and $\boldsymbol{X}$, possibly under a rank constraint. Assuming elements in $\boldsymbol{W}$ are iid Gaussian with variance σ_w^2, then

$$p(\boldsymbol{Y} | \boldsymbol{A}, \boldsymbol{X}) = \prod_{n=1}^N \prod_{m=1}^M \Phi\Big(\frac{y_{nm} z_{nm}}{\sigma_w}\Big), \tag{1.12}$$

where $\Phi(\cdot)$ is the standard normal CDF and $z_{nm} = [\boldsymbol{A}\boldsymbol{X}]_{nm}$. Then, one could solve

$$\Big\{\widehat{\boldsymbol{A}}, \widehat{\boldsymbol{X}}\Big\} = \underset{\boldsymbol{A}, \boldsymbol{X}}{\arg\max} \log p(\boldsymbol{Y} | \boldsymbol{A}, \boldsymbol{X}) \quad \text{s.t.} \quad \operatorname{rank}(\boldsymbol{A}\boldsymbol{X}) \leq R. \tag{1.13}$$

1.1.4 Summary

So far, we have introduced linear and generalized linear models, with various examples. We have also introduced the ML and MAP inference frameworks, where one first has to specify $p(\boldsymbol{y}|\boldsymbol{z})$. Then, one may specify a regularization $\lambda g(\boldsymbol{x})$, which corresponds to the prior $p(\boldsymbol{x}) \propto \exp\Big(\lambda g(\boldsymbol{x})\Big)$. Combining these, one can solve

$$\widehat{\boldsymbol{x}}_{\mathsf{MAP}} = \arg\max_{\boldsymbol{x}} \Big\{ \log p(\boldsymbol{y}|\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}) + \lambda g(\boldsymbol{x}) \Big\}. \tag{1.14}$$

There are advantages and disadvantages of this approach. Problem (1.14) is interpretable and convex for many choices of $p(\boldsymbol{y}|\boldsymbol{z})$ and $g(\boldsymbol{x})$. However, iterative methods to solve Problem (1.14) may require many iterations to converge, leading to a large computational cost of this approach. This high computational cost is exacerbated when $\boldsymbol{A}$ is large because each iteration is more costly. Additionally, the complexity is further increased when λ and hyperparameters associated with $p(\boldsymbol{y}|\boldsymbol{z})$ must be tuned, where cross-validation (CV) is the standard approach and involves solving Problem (1.14) with multiple subsets of $\boldsymbol{A}$ and multiple hypothesized values of λ (and any other hyperparameter). Moreover, Problem (1.14) is just one type of estimation. There are others, such as minimum mean square error (MMSE) that may be more desirable for some problems, but are in general not as numerically tractable as MAP estimation. In the next section we introduce a class of inference algorithms based on Approximate Message Passing that address some of these issues.

1.2 Introduction to Approximate Message Passing

In this section we introduce Approximate Message Passing (AMP), which are a class of algorithms well suited to inference in the linear and generalized linear models.

To address the issues mentioned in Section 1.1.4, approximately a decade ago, the original AMP algorithm of Donoho, Maleki, and Montanari [1] was created for inference with the standard linear model in (1.1). AMP assumes elements in $\boldsymbol{A}$ are iid sub-Gaussian, and requires a separable prior on $\boldsymbol{x}$, i.e., $p_{\mathsf{x}}(\boldsymbol{x}) = \prod_{n=1}^{N} p_{\mathsf{x}}(x_n)$. The AMP algorithm then uses various approximations based on the central limit theorem and Taylor series to approximate MAP or MMSE inference by approximating the sum-product (SPA) or min-sum (MSA) loopy belief propagation algorithms, respectively, on the factor graph associated with (1.1) and $p_{\mathsf{x}}(\boldsymbol{x})$. AMP was originally applied to the LASSO problem in (1.5) by setting $\log p_{\mathsf{x}}(x_n) = -\lambda|x_n| + \text{const.}$, where it was demonstrated to converge in substantially fewer iterations than existing state-of-that-art algorithms.

However, AMP is restrictive due to its AWGN-only assumption. Subsequently, AMP was extended by Rangan [2] to generalized linear models, yielding the Generalized AMP (GAMP) algorithm. GAMP still requires separable $p_{\mathsf{x}}(\boldsymbol{x})$, and also requires $p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y}|\boldsymbol{z}) = \prod_{m=1}^{M} p_{\mathsf{y}|\mathsf{z}}(y_m|z_m)$, which combine to form the posterior

$$p_{\mathsf{x}|\mathsf{y}}(\boldsymbol{x}|\boldsymbol{y}) \propto \prod_{m=1}^{M} p_{\mathsf{y}|\mathsf{z}}(y_m|z_m) \prod_{n=1}^{N} p_{\mathsf{x}}(x_n). \tag{1.15}$$

AMP and GAMP have several advantages over conventional techniques for inference in the standard and generalized linear models. First, both AMP and GAMP give accurate approximations of the SPA and MSA under large i.i.d. sub-Gaussian $\boldsymbol{A}$, while maintaining a computational complexity of only $O(MN)$. Through numerical simulations, both AMP and GAMP have been demonstrated to be superior to existing state-of-the-art inference techniques for a wide variety of applications. Furthermore, both can be rigorously analyzed via the state-evolution framework, which proves that they compute MMSE optimal estimates of $\boldsymbol{x}$ in certain regimes [3]. Finally, the

GAMP algorithm can be readily combined with the expectation-maximization (EM) algorithm to tune hyperparameters associated with $p_{\mathsf{y}|\mathsf{z}}$ and p_{x} online, avoiding the need to perform expensive cross-validation [4].

A limitation of AMP [1] and GAMP [2] is that they treat only problems with i.i.d. estimand $\boldsymbol{x}$ and separable, scalar, likelihood $p(\boldsymbol{y}|\boldsymbol{z}) = \prod_{m=1}^{M} p(y_m|z_m)$. Thus, *Hybrid GAMP* (HyGAMP) [5] was developed to tackle problems with a structured prior and/or likelihood. Specifically, HyGAMP allows structure on $p_{\mathsf{y}|\mathsf{z}}$ first mentioned in Section 1.1.2, and allows structure on rows of $\boldsymbol{X}$, provided by the prior $p_{\mathsf{x}}(\boldsymbol{x}_n)$ i.e., HyGAMP assumes the probabilistic model

$$p_{\mathsf{X}|\mathsf{y}}(\boldsymbol{X}|\boldsymbol{y}) = \prod_{m=1}^{M} p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m) \prod_{n=1}^{N} p_{\mathsf{x}}(\boldsymbol{x}_n), \tag{1.16}$$

where $p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m) \neq \prod_{k=1}^{K} p_{\mathsf{y}|\mathsf{z}}(y_m|z_{mk})$, and $\boldsymbol{x}_n^{\mathsf{T}}$ is the nth row of $\boldsymbol{X}$ (so, for example, a prior could be constructed to encourage row-sparsity in $\boldsymbol{X}$). The HyGAMP algorithm will be described in more detail in Chapter 2.

Finally, AMP has also been extended to generalized bilinear inference, for example in the BiGAMP [6] and LowRAMP [7,8] algorithms, both of which assume the model

$$p_{\mathsf{X}|\mathsf{Y}}(\boldsymbol{X}|\boldsymbol{Y}) \propto \left[\prod_{m=1}^{M}\prod_{k=1}^{K} p_{\mathsf{y}|\mathsf{z}}(y_{mk}|z_{mk})\right]\left[\prod_{n=1}^{N}\prod_{k=1}^{K} p_{\mathsf{x}}(x_{nk})\right]\left[\prod_{m=1}^{M}\prod_{n=1}^{N} p_{\mathsf{a}}(a_{mn})\right], \tag{1.17}$$

but differ in their specific approximation to the sum-product algorithm.

1.3 Outline and Contributions

An outline of this dissertation and our research contributions are summarized here.

1.3.1 The HyGAMP Algorithm

First, in Chapter 2 we present the HyGAMP algorithm [5], which is an extension of the GAMP algorithm to the structured $p(y_m|\boldsymbol{z}_m)$ presented in Section 1.1.2. Similar to GAMP, the HyGAMP algorithm comes in two flavors: SPA-HyGAMP and MSA-HyGAMP, which approximate MMSE and MAP estimation, respectively. Then, we explain the "simplified" version of the HyGAMP algorithm, first presented in [9], (SHyGAMP), which drastically reduces the computational complexity of its implementation and allows this approach to be computationally competitive with existing state-of-the-art approaches for various applications.

1.3.2 Sparse Multinomial Logistic Regression

In Chapter 3 we apply the SPA and MSA HyGAMP algorithms from Chapter 2 to the sparse multinomial logistic regression (SMLR) problem [10]. In this problem we are given training data consisting of feature-vector, label pairs $\{\boldsymbol{a}_m, y_m\}_{m=1}^{M}$, $\boldsymbol{a}_m \in \mathbb{R}^N$, $y_m \in \{1, ..., K\}$ with the goal of designing a weight matrix $\widehat{\boldsymbol{X}} \in \mathbb{R}^{N \times K}$ that accurately predicts the class label $\widehat{y}_0$ on the unlabeled feature vector $\boldsymbol{a}_0$, via $\widehat{y}_0 = \arg\max_k [\widehat{\boldsymbol{X}}^\mathsf{T} \boldsymbol{a}_0]_k$. The multinomial logistic regression approach to this problem models the probability that $\boldsymbol{a}_m$ belongs to class y_m via

$$p(y_m|\boldsymbol{z}_m) = \frac{\exp\Big([\boldsymbol{z}_m]_{y_m}\Big)}{\sum_{k=1}^{K} \exp\Big([\boldsymbol{z}_m]_k\Big)}, \tag{1.18}$$

where $\boldsymbol{z}_m = \boldsymbol{X}^\mathsf{T} \boldsymbol{a}_m$. HyGAMP can be applied to this problem by using (1.18) in (1.15). We focus on the regime where $M < N$ and therefore use a sparsity-promoting prior $p(\boldsymbol{X})$. We note that while the MSA-HyGAMP algorithm agrees with the standard MAP estimation technique [10] for this problem, we show that the SPA-HyGAMP approach can be interpreted as the test-error-rate-minimizing approach to designing the weight matrix $\widehat{\boldsymbol{X}}$. Then, we extend both the SPA and MSA HyGAMP algorithms to their SHyGAMP counterparts, after which we show through extensive numerical experiments that our approach surpasses existing state-of-the-art approaches to the SMLR problem in both accuracy and computational complexity.

1.3.3 Sketched Clustering

In Chapter 4 we apply the SPA-SHyGAMP algorithm from Chapter 2 to the sketched clustering problem [11,12]. The sketched clustering problem is a variation of the traditional clustering problem, where one is given a dataset $\boldsymbol{D} \in \mathbb{R}^{N \times T}$ comprising of T feature-vectors of dimension N, and wants to find K centroids $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_K] \in \mathbb{R}^{N \times K}$ that minimize the sum of squared errors (SSE), where

$$\text{SSE}(\boldsymbol{X}, \boldsymbol{D}) = \frac{1}{T} \sum_{t=1}^{T} \min_k \|\boldsymbol{d}_t - \boldsymbol{x}_k\|_2^2. \tag{1.19}$$

The sketched clustering approach to this problem first "sketches" the data matrix $\boldsymbol{D}$ into a relatively low dimensional vector $\boldsymbol{y}$ via a non-linear transformation. Then, a sketched clustering recovery algorithm attempts to extract $\boldsymbol{X}$ from $\boldsymbol{y}$ instead of $\boldsymbol{D}$.

In Chapter 4, with our particular choice of sketching function, we show how we are able to apply the SPA-SHyGAMP algorithm to this problem. We then show through numerical experiments that our approach, in some regimes, has better accuracy, computational complexity, and memory complexity than k-means, and has better time and sample complexity than the only other known sketched clustering algorithm, CL-OMPR [12].

1.3.4 Adaptive Detection from Quantized Measurements

Finally, in Chapter 5 we shift gears and investigate adaptive detection using quantized measurements. First, we study the effects of using unquantized detection techniques but with quantized measurements. We observe that there is not a significant loss to using quantized measurements, as long as the interference power is small. However, when the interference power is large, the lack of dynamic range in the quantizer eliminates any chance for reliable detection. We then investigate and develop several techniques to improve detection performance in the high-interference regime. First, we consider two basic techniques that are commonly used in conjunction with quantized measurements: dithering and companding. Then, we develop and apply the generalized likelihood ratio test (GLRT) to this problem using the appropriate quantization and noise model, which results in a generalized linear model. We observe this approach does not offer significant improvement, so we conclude with a variety of proposed techniques that are based upon applying an interference-reduction transformation to the data in the analog domain prior to quantization.

Chapter 2: The Hybrid-GAMP Algorithm

In this Chapter, we present a special case[1] of the hybrid generalized approximate message passing (HyGAMP) algorithm [5] that will later be used to tackle the multinomial logistic regression problem in Chapter 3 and the sketched clustering problem in Chapter 4. The HyGAMP algorithm is an extension of the AMP and GAMP algorithms originally introduced in Section 1.2.

2.1 Model

HyGAMP assumes the probabilistic model

$$p_{\mathsf{X}|\mathsf{y}}(\boldsymbol{X}|\boldsymbol{y}) \propto \prod_{m=1}^{M} p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m) \prod_{n=1}^{N} p_{\mathsf{x}}(\boldsymbol{x}_n), \tag{2.1}$$

where $\boldsymbol{x}_n^{\mathsf{T}}$ is the nth row of $\boldsymbol{X} \in \mathbb{R}^{N\times K}$ and $\boldsymbol{z}_m^{\mathsf{T}}$ is the mth row of $\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{X}$, where $\boldsymbol{A} \in \mathbb{R}^{M\times N}$. Under (2.1), the HyGAMP algorithm approximates either

$$\widehat{\boldsymbol{X}}_{\mathsf{MMSE}} \triangleq \mathrm{E}\{\boldsymbol{X}|\boldsymbol{y}\} \tag{2.2}$$

or

$$\widehat{\boldsymbol{X}}_{\mathsf{MAP}} \triangleq \arg\max_{\boldsymbol{X}} \log p_{\mathsf{X}|\mathsf{y}}(\boldsymbol{X}|\boldsymbol{y}) \tag{2.3}$$

[1]In particular, the "hybrid" in HyGAMP refers to combining approximate message passing with exact message passing. However, we do not use that feature of HyGAMP and therefore do not present it for simplicity. The version of HyGAMP that we present is a simple generalization of the GAMP algorithm (i.e., GAMP and HyGAMP coincide when $K = 1$). The "hybrid" modifier is therefore unnecessary, but we retain it to remain consistent with previously published works.

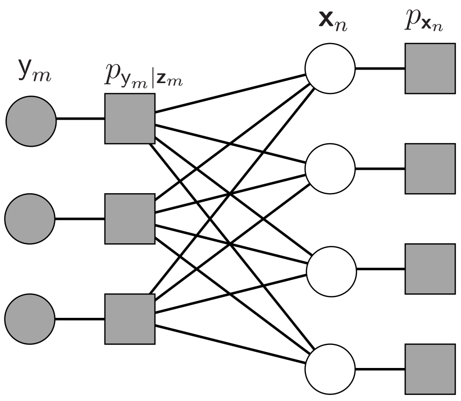


Figure 2.1: Factor graph representations of (2.1), with white/gray circles denoting unobserved/observed random variables, and gray rectangles denoting pdf "factors".

by approximating the sum-product (SPA) or min-sum (MSA) loopy belief propagation algorithms, respectively, on the factor graph associated with (2.1), which is shown in Figure 2.1. Throughout this document we may use the terms SPA-HyGAMP and MMSE-HyGAMP interchangeably, likewise with MSA/MAP HyGAMP.

2.2 The HyGAMP Algorithm

The HyGAMP algorithm is presented in Table 2.1. It requires matrix $\boldsymbol{A}$ and measurement vector $\boldsymbol{y}$, we well as the distributions $p_{\mathsf{x}|\mathsf{r}}$ and $p_{\mathsf{z}|\mathsf{y},\mathsf{p}}$, which are given in (2.4) and (2.5) and depend on $p_{\mathsf{x}}(\boldsymbol{x}_n)$ and $p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)$, respectively. It also requires initialization $\widehat{\boldsymbol{x}}_n(0)$ and $\boldsymbol{Q}^{\mathsf{x}}_n(0)$; we note the specific choice of initialization depends on the application and more details will be given at the appropriate time.

Observe that HyGAMP breaks the $N \times K$ dimensional inference problem into a sequence of K-dimensional inference problems, which are shown in Lines 5-9 and 16-20 of Table 2.1. Not surprisingly, in the MAP case, the K-dimensional inference problems involve MAP estimation, while in the MMSE case the inference problems require

MMSE estimation. They involve the following approximate posterior distributions on $\boldsymbol{x}_n$ and $\boldsymbol{z}_m$, respectively:

$$p_{\mathsf{x}|\mathsf{r}}(\boldsymbol{x}_n|\widehat{\boldsymbol{r}}_n;\boldsymbol{Q}^{\mathsf{r}}_n) = \frac{p_{\mathsf{x}}(\boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}^{\mathsf{r}}_n)}{\int p_{\mathsf{x}}(\boldsymbol{x}'_n)\mathcal{N}(\boldsymbol{x}'_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}^{\mathsf{r}}_n)\,\mathrm{d}\boldsymbol{x}'_n} \tag{2.4}$$

and

$$p_{\mathsf{z}|\mathsf{y},\mathsf{p}}(\boldsymbol{z}_m|y_m,\widehat{\boldsymbol{p}}_m;\boldsymbol{Q}^{\mathsf{p}}_m) = \frac{p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}_m)}{\int p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}'_m)\mathcal{N}(\boldsymbol{z}'_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}_m)\,\mathrm{d}\boldsymbol{z}'_m}. \tag{2.5}$$

These distributions depend on the choice of $p_{\mathsf{x}}(\boldsymbol{x}_n)$ and $p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)$. As we will see in future chapters, these low-dimensional inference problems may be non-trivial to solve for certain choices of $p_{\mathsf{x}}(\boldsymbol{x}_n)$ and $p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)$.

In the large system limit with iid Gaussian $\boldsymbol{A}$, $\widehat{\boldsymbol{r}}_n$ can be interpreted as a Gaussian noise corrupted version of the true $\boldsymbol{x}_n$ with noise distribution $\mathcal{N}(\boldsymbol{0},\boldsymbol{Q}^{\mathsf{r}}_n)$, and similarly $\widehat{\boldsymbol{p}}_m$ as a Gaussian noise corrupted version of the true $\boldsymbol{z}_m$ with noise distribution $\mathcal{N}(\boldsymbol{0},\boldsymbol{Q}^{\mathsf{p}}_m)$. Note that in many applications, $\boldsymbol{A}$ is not iid Gaussian, but treating it as such appears to work sufficiently well.

2.3 Simplified HyGAMP

Excluding the inference steps in lines 5-9 and 16-20 of Table 2.1, the computational complexity of HyGAMP is $O\Big(MNK^2 + (M+N)K^3\Big)$, where the "$MNK^2$" term is from line 2, and the "$(M+N)K^3$" term is from lines 11 and 13, where it requires computing $O(M+N)$ $K\times K$ matrix inversions at every iteration. The computational complexity of the inference steps are excluded because they depend on $p_{\mathsf{y}|\mathsf{z}}$ and p_{x}. However, they may be complicated due to both the form of $p_{\mathsf{y}|\mathsf{z}}$ and p_{x} and full covariance matrices $\boldsymbol{Q}^{\mathsf{x}}_n$, $\boldsymbol{Q}^{\mathsf{r}}_n$, etc.

```
Require: Mode ∈ {SPA, MSA}, matrix $\boldsymbol{A}$, vector $\boldsymbol{y}$, pdfs $p_{\mathsf{x}|\mathsf{r}}$ and $p_{\mathsf{z}|\mathsf{y},\mathsf{p}}$ from (2.4)-(2.5),
    initializations $\widehat{\boldsymbol{x}}_n(0)$, $\boldsymbol{Q}^{\mathsf{x}}_n(0)$.
Ensure: $t \leftarrow 0$; $\widehat{\boldsymbol{s}}_m(0) \leftarrow \boldsymbol{0}$.
 1: repeat
 2:   $\forall m : \boldsymbol{Q}^{\mathsf{p}}_m(t) \leftarrow \sum_{n=1}^N A_{mn}^2 \boldsymbol{Q}^{\mathsf{x}}_n(t)$
 3:   $\forall m : \widehat{\boldsymbol{p}}_m(t) \leftarrow \sum_{n=1}^N A_{mn} \widehat{\boldsymbol{x}}_n(t) - \boldsymbol{Q}^{\mathsf{p}}_m(t)\widehat{\boldsymbol{s}}_m(t)$
 4:   if MSA then {for $m = 1 \dots M$}
 5:     $\widehat{\boldsymbol{z}}_m(t) \leftarrow \arg\max_{\boldsymbol{z}} \log p_{\mathsf{z}|\mathsf{y},\mathsf{p}}\Big(\boldsymbol{z}_m \Big| y_m, \widehat{\boldsymbol{p}}_m(t); \boldsymbol{Q}^{\mathsf{p}}_m(t)\Big)$
 6:     $\boldsymbol{Q}^{\mathsf{z}}_m(t) \leftarrow \Big[-\frac{\partial^2}{\partial \boldsymbol{z}^2} \log p_{\mathsf{z}|\mathsf{y},\mathsf{p}}\Big(\widehat{\boldsymbol{z}}_m(t) \Big| y_m, \widehat{\boldsymbol{p}}_m(t); \boldsymbol{Q}^{\mathsf{p}}_m(t)\Big)\Big]^{-1}$
 7:   else if SPA then {for $m = 1 \dots M$}
 8:     $\widehat{\boldsymbol{z}}_m(t) \leftarrow \mathrm{E}\Big\{\mathbf{z}_m \Big| y_m, \mathbf{p}_m = \widehat{\boldsymbol{p}}_m(t); \boldsymbol{Q}^{\mathsf{p}}_m(t)\Big\}$
 9:     $\boldsymbol{Q}^{\mathsf{z}}_m(t) \leftarrow \mathrm{Cov}\Big\{\mathbf{z}_m \Big| y_m, \mathbf{p}_m = \widehat{\boldsymbol{p}}_m(t); \boldsymbol{Q}^{\mathsf{p}}_m(t)\Big\}$
10:   end if
11:   $\forall m : \boldsymbol{Q}^{\mathsf{s}}_m(t) \leftarrow [\boldsymbol{Q}^{\mathsf{p}}_m(t)]^{-1} - [\boldsymbol{Q}^{\mathsf{p}}_m(t)]^{-1}\boldsymbol{Q}^{\mathsf{z}}_m(t)[\boldsymbol{Q}^{\mathsf{p}}_m(t)]^{-1}$
12:   $\forall m : \widehat{\boldsymbol{s}}_m(t+1) \leftarrow [\boldsymbol{Q}^{\mathsf{p}}_m(t)]^{-1}\Big(\widehat{\boldsymbol{z}}_m(t) - \widehat{\boldsymbol{p}}_m(t)\Big)$
13:   $\forall n : \boldsymbol{Q}^{\mathsf{r}}_n(t) \leftarrow \Big[\sum_{m=1}^M A_{mn}^2 \boldsymbol{Q}^{\mathsf{s}}_m(t)\Big]^{-1}$
14:   $\forall n : \widehat{\boldsymbol{r}}_n(t) \leftarrow \widehat{\boldsymbol{x}}_n(t) + \boldsymbol{Q}^{\mathsf{r}}_n(t) \sum_{m=1}^M A_{mn} \widehat{\boldsymbol{s}}_m(t+1)$
15:   if MSA then {for $n = 1 \dots N$}
16:     $\widehat{\boldsymbol{x}}_n(t+1) \leftarrow \arg\max_{\boldsymbol{x}} \log p_{\mathsf{x}|\mathsf{r}}\Big(\boldsymbol{x}_n \Big| \widehat{\boldsymbol{r}}_n(t); \boldsymbol{Q}^{\mathsf{r}}_n(t)\Big)$
17:     $\boldsymbol{Q}^{\mathsf{x}}_n(t+1) \leftarrow \Big[-\frac{\partial^2}{\partial \boldsymbol{x}^2} \log p_{\mathsf{x}|\mathsf{r}}\Big(\widehat{\boldsymbol{x}}_n(t+1) \Big| \widehat{\boldsymbol{r}}_n(t); \boldsymbol{Q}^{\mathsf{r}}_n(t)\Big)\Big]^{-1}$
18:   else if SPA then {for $n = 1 \dots N$}
19:     $\widehat{\boldsymbol{x}}_n(t+1) \leftarrow \mathrm{E}\Big\{\mathbf{x}_n \Big| \mathbf{r}_n = \widehat{\boldsymbol{r}}_n(t); \boldsymbol{Q}^{\mathsf{r}}_n(t)\Big\}$
20:     $\boldsymbol{Q}^{\mathsf{x}}_n(t+1) \leftarrow \mathrm{Cov}\Big\{\mathbf{x}_n \Big| \mathbf{r}_n = \widehat{\boldsymbol{r}}_n(t); \boldsymbol{Q}^{\mathsf{r}}_n(t)\Big\}$
21:   end if
22:   $t \leftarrow t + 1$
23: until Terminated
```

Table 2.1: The HyGAMP Algorithm. For clarity, note that all matrices $\boldsymbol{Q}^{\mathsf{x}}_n$, $\boldsymbol{Q}^{\mathsf{z}}_m$, etc are $K \times K$, and vectors $\widehat{\boldsymbol{x}}_n$, $\widehat{\boldsymbol{z}}_m$, $\widehat{\boldsymbol{p}}_m$, etc are $K \times 1$.

For our work in sparse multinomial logistic regression and sketched clustering, we proposed a simplified HyGAMP (SHyGAMP) in order to be computationally competitive with existing state-of-the-art algorithms for those applications. In SHyGAMP, we simply force all covariance matrices in Table 2.1, e.g., $\boldsymbol{Q}^{\mathsf{x}}_n$, $\boldsymbol{Q}^{\mathsf{p}}_m$, etc, to be diagonal, e.g., $\boldsymbol{Q}^{\mathsf{x}}_n = \text{diag}\{q^{\mathsf{x}}_{n1}, ..., q^{\mathsf{x}}_{nK}\}$. In many applications we have found when we compared SHyGAMP to HyGAMP that SHyGAMP had negligible loss in accuracy, while having a drastic decrease in computational complexity. In particular, the computational complexity of SHyGAMP (excluding the inference steps) is $O(MNK)$. Furthermore, for many choices of $p_{\mathsf{y}|\mathsf{z}}$ and p_{x}, using diagonal covariance matrices $\boldsymbol{Q}^{\mathsf{x}}_n$, $\boldsymbol{Q}^{\mathsf{r}}_n$, etc decreases the computational complexity of the inference steps.

2.4 Scalar-variance Approximation

We further approximate the SHyGAMP algorithm using the *scalar variance* GAMP approximation from [13], which reduces the memory and complexity of the algorithm. The scalar variance approximation first approximates the variances $\{q^{\mathsf{x}}_{nk}\}$ by a value invariant to both n and k, i.e.,

$$q^{\mathsf{x}} \triangleq \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} q^{\mathsf{x}}_{nk}. \tag{2.6}$$

Then, in line 2 in Table 2.1, we use the approximation

$$q^{\mathsf{p}}_{mk} \approx \sum_{n=1}^{N} A^2_{mn} q^{\mathsf{x}} \overset{(a)}{\approx} \frac{\|\boldsymbol{A}\|^2_F}{M} q^{\mathsf{x}} \triangleq q^{\mathsf{p}}. \tag{2.7}$$

The approximation (a), after precomputing $\|\boldsymbol{A}\|^2_F$, reduces the complexity of line 2 from $O(NK)$ to $O(1)$. We next define

$$q^{\mathsf{s}} \triangleq \frac{1}{MK} \sum_{m=1}^{M} \sum_{k=1}^{K} q^{\mathsf{s}}_{mk} \tag{2.8}$$

and in line 13 we use the approximation

$$q^{\mathsf{r}}_{nk} \approx \left(\sum_{m=1}^{M} A^2_{mn} q^{\mathsf{s}} \right)^{-1} \approx \frac{N}{q^{\mathsf{s}} \|\boldsymbol{A}\|^2_F} \triangleq q^{\mathsf{r}}. \tag{2.9}$$

The complexity of line 13 then simplifies from $O(MK)$ to $O(1)$. For clarity, we note that after applying the scalar variance approximation, we have $\boldsymbol{Q}_n^{\mathsf{x}} = q^{\mathsf{x}}\boldsymbol{I}_K \,\forall\, n$, and similar for $\boldsymbol{Q}_n^{\mathsf{r}}$, $\boldsymbol{Q}_m^{\mathsf{p}}$ and $\boldsymbol{Q}_m^{\mathsf{z}}$.

2.5 Conclusion

The HyGAMP algorithm is an extension of the GAMP algorithm to inference problems with additional structure on $p_{\mathsf{y}|\mathsf{z}}$, notably the case where $p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m) \neq \prod_{k=1}^{K} p_{\mathsf{y}|\mathsf{z}}(y_m|z_{mk})$. The HyGAMP algorithm can be applied to many inference problems by appropriately selecting p_{x} and $p_{\mathsf{y}|\mathsf{z}}$. However, as we will see in future chapters, for many choices of p_{x} and $p_{\mathsf{y}|\mathsf{z}}$, the inference problems in Lines 5-9 and 16-20 of Table 2.1 are non-trivial to compute. Moreover, the HyGAMP algorithm is complicated by the full covariance matrices $\boldsymbol{Q}^{\mathsf{x}}$, etc, and so for practical considerations a simplified version, SHyGAMP, is proposed. In Chapters 3 and 4, we apply the SHyGAMP algorithm to the problems or sparse multinomial logistic regression and sketched clustering, respectively.

Chapter 3: Sparse Multinomial Logistic Regression via Approximate Message Passing

3.1 Introduction

In this chapter[2], we consider the problems of multiclass (or polytomous) linear classification and feature selection. In both problems, one is given training data of the form $\{(y_m, \boldsymbol{a}_m)\}_{m=1}^M$, where $\boldsymbol{a}_m \in \mathbb{R}^N$ is a vector of features and $y_m \in \{1, \ldots, K\}$ is the corresponding K-ary class label. In *multiclass classification*, the goal is to infer the unknown label y_0 associated with a newly observed feature vector $\boldsymbol{a}_0$. In the *linear* approach to this problem, the training data are used to design a weight matrix $\boldsymbol{X} \in \mathbb{R}^{N \times K}$ that generates a vector of "scores" $\boldsymbol{z}_0 \triangleq \boldsymbol{X}^\mathsf{T} \boldsymbol{a}_0 \in \mathbb{R}^K$, the largest of which can be used to predict the unknown label, i.e.,

$$\widehat{y}_0 = \arg\max_k [\boldsymbol{z}_0]_k. \tag{3.1}$$

In *feature selection*, the goal is to determine which *subset* of the N features $\boldsymbol{a}_0$ is needed to accurately predict the label y_0.

[2]Work presented in this chapter is largely excerpted from a journal publication co-authored with Philip Schniter, titled "Sparse Multinomial Logistic Regression via Approximate Message Passing" [9]. We note that a preliminary version of this work was published in the Master's Thesis authored by Evan Byrne [14]. It is included here as well to include changes made during the peer-review process, and because it forms part of a larger story when combined with the other chapters of this dissertation.

We are particularly interested in the setting where the number of features, N, is large and greatly exceeds the number of training examples, M. Such problems arise in a number of important applications, such as micro-array gene expression [15, 16], multi-voxel pattern analysis (MVPA) [17, 18], text mining [19, 20], and analysis of marketing data [21].

In the $N \gg M$ case, accurate linear classification and feature selection may be possible if the labels are influenced by a sufficiently small number, S, of the total N features. For example, in binary linear classification, performance guarantees are possible with only $M = O(S \log N/S)$ training examples when $\boldsymbol{a}_m$ is i.i.d. Gaussian [22].

Note that, when $S \ll N$, accurate linear classification can be accomplished using a *sparse* weight matrix $\boldsymbol{X}$, i.e., a matrix where all but a few rows are zero-valued.

3.1.1 Multinomial logistic regression

For multiclass linear classification and feature selection, we focus on the approach known as *multinomial logistic regression* (MLR) [23], which can be described using a generative probabilistic model. Here, the label vector $\boldsymbol{y} \triangleq [y_1, \dots, y_M]^\mathsf{T}$ is modeled as a realization of a random[3] vector $\mathsf{y} \triangleq [\mathsf{y}_1, \dots, \mathsf{y}_M]^\mathsf{T}$, the "true" weight matrix $\boldsymbol{X}$ is modeled as a realization of a random matrix X, and the features $\boldsymbol{A} \triangleq [\boldsymbol{a}_1, \dots, \boldsymbol{a}_M]^\mathsf{T}$ are treated as deterministic. Moreover, the labels y_m are modeled as conditionally independent given the scores $\mathsf{z}_m \triangleq \mathsf{X}^\mathsf{T} \boldsymbol{a}_m$, i.e.,

$$\Pr\{\mathsf{y} = \boldsymbol{y} \,|\, \mathsf{X} = \boldsymbol{X}; \boldsymbol{A}\} = \prod_{m=1}^{M} p_{\mathsf{y}|\mathsf{z}}(y_m | \boldsymbol{X}^\mathsf{T} \boldsymbol{a}_m), \tag{3.2}$$

[3]For clarity, we typeset random quantities in sans-serif font and deterministic quantities in serif font.

and distributed according to the multinomial logistic (or soft-max) pmf:

$$p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m) = \frac{\exp([\boldsymbol{z}_m]_{y_m})}{\sum_{k=1}^{K}\exp([\boldsymbol{z}_m]_k)}, \quad y_m \in \{1,\dots,K\}. \tag{3.3}$$

The rows $\mathbf{x}_n^\mathsf{T}$ of the weight matrix $\mathbf{X}$ are then modeled as i.i.d.,

$$p_{\mathbf{X}}(\boldsymbol{X}) = \prod_{n=1}^{N} p_{\mathsf{x}}(\boldsymbol{x}_n), \tag{3.4}$$

where p_{x} may be chosen to promote sparsity.

3.1.2 Existing methods

Several sparsity-promoting MLR algorithms have been proposed (e.g., [10,24–28]), differing in their choice of p_{x} and methodology of estimating $\mathbf{X}$. For example, [10,25, 26] use the i.i.d. Laplacian prior

$$p_{\mathsf{x}}(\boldsymbol{x}_n;\lambda) = \prod_{k=1}^{K}\frac{\lambda}{2}\exp(-\lambda|x_{nk}|), \tag{3.5}$$

with λ tuned via cross-validation. To circumvent this tuning problem, [27] employs the Laplacian scale mixture

$$p_{\mathsf{x}}(\boldsymbol{x}_n) = \prod_{k=1}^{K}\int\left[\frac{\lambda}{2}\exp(-\lambda|x_{nk}|)\right]p(\lambda)\,\mathrm{d}\lambda, \tag{3.6}$$

with Jeffrey's non-informative hyperprior $p(\lambda) \propto \frac{1}{\lambda}1_{\lambda\geq 0}$. The relevance vector machine (RVM) approach [24] uses the Gaussian scale mixture

$$p_{\mathsf{x}}(\boldsymbol{x}_n) = \prod_{k=1}^{K}\int\mathcal{N}(x_{nk};0,\nu)p(\nu)\,\mathrm{d}\nu, \tag{3.7}$$

with inverse-gamma $p(\nu)$ (i.e., the conjugate hyperprior), resulting in an i.i.d. student's t distribution for p_{x}. However, other choices are possible. For example, the exponential hyperprior $p(\nu;\lambda) = \frac{\lambda^2}{2}\exp(-\frac{\lambda^2}{2}\nu)1_{\nu\geq 0}$ would lead back to the i.i.d. Laplacian distribution (3.5) for p_{x} [29]. Finally, [28] uses

$$p_{\mathsf{x}}(\boldsymbol{x}_n;\lambda) \propto \exp(-\lambda\|\boldsymbol{x}_n\|_2), \tag{3.8}$$

which encourages row-sparsity in $\boldsymbol{X}$.

Once the probabilistic model (3.2)-(3.4) has been specified, a procedure is needed to infer the weights $\mathbf{X}$ from the training data $\{(y_m, \boldsymbol{a}_m)\}_{m=1}^{M}$. The Laplacian-prior methods [10, 25, 26, 28] use the maximum a posteriori (MAP) estimation framework:

$$\widehat{\boldsymbol{X}} = \arg\max_{\boldsymbol{X}} \log p(\boldsymbol{X}|\boldsymbol{y};\boldsymbol{A}) \tag{3.9}$$

$$= \arg\max_{\boldsymbol{X}} \sum_{m=1}^{M} \log p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{X}^{\mathsf{T}}\boldsymbol{a}_m) + \sum_{n=1}^{N} \log p_{\mathsf{x}}(\boldsymbol{x}_n), \tag{3.10}$$

where Bayes' rule was used for (3.10). Under p_{x} from (3.5) or (3.8), the second term in (3.10) reduces to $-\lambda\sum_{n=1}^{N}\|\boldsymbol{x}_n\|_1$ or $-\lambda\sum_{n=1}^{N}\|\boldsymbol{x}_n\|_2$, respectively. In this case, (3.10) is concave and can be maximized in polynomial time; [10, 25, 26, 28] employ (block) coordinate ascent for this purpose. The papers [24] and [27] handle the scale-mixture priors (3.6) and (3.7), respectively, using the evidence maximization framework [30]. This approach yields a double-loop procedure: the hyperparameter λ or ν is estimated in the outer loop, and—for fixed λ or ν—the resulting concave (i.e., ℓ_2 or ℓ_1 regularized) MAP optimization is solved in the inner loop.

The methods [10,24–28] described above all yield a sparse point estimate $\widehat{\boldsymbol{X}}$. Thus, feature selection is accomplished by examining the row-support of $\widehat{\boldsymbol{X}}$ and classification is accomplished through (3.1).

3.1.3 Contributions

In Section 3.2, we propose new approaches to sparse-weight MLR based on the *hybrid generalized approximate message passing* (HyGAMP) framework from [13]. HyGAMP offers tractable approximations of the sum-product and min-sum message passing algorithms [31] by leveraging results of the central limit theorem that hold in the large-system limit: $\lim_{N,M\to\infty}$ with fixed N/M. Without approximation, both the sum-product algorithm (SPA) and min-sum algorithm (MSA) are intractable due to the forms of $p_{\mathsf{y}|\mathsf{z}}$ and p_{x} in our problem.

For context, we note that HyGAMP is a generalization of the original GAMP approach from [2], which cannot be directly applied to the MLR problem because the likelihood function (3.3) is not separable, i.e., $p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m) \neq \prod_k p(y_m|z_{mk})$. GAMP can, however, be applied to *binary* classification and feature selection, as in [32]. Meanwhile, GAMP is itself a generalization of the original AMP approach from [1,33], which requires $p_{\mathsf{y}|\mathsf{z}}$ to be both separable and Gaussian.

With the HyGAMP algorithm from [13], message passing for sparse-weight MLR reduces to an iterative update of $O(M+N)$ multivariate Gaussian pdfs, each of dimension D. Although HyGAMP makes MLR tractable, it is still not computationally practical for the large values of M and N in contemporary applications (e.g., $N \sim 10^4$ to 10^6 in genomics and MVPA). Similarly, the non-conjugate variational message passing technique from [34] requires the update of $O(MN)$ multivariate Gaussian pdfs of dimension D, which is even less practical for large M and N.

Thus, in Section 3.3, we propose a simplified HyGAMP (SHyGAMP) algorithm for MLR that approximates HyGAMP's mean and variance computations in an efficient manner. In particular, we investigate approaches based on numerical integration, importance sampling, Taylor-series approximation, and a novel Gaussian-mixture approximation, and we conduct numerical experiments that suggest the superiority of the latter.

In Section 3.4, we detail two approaches to tune the hyperparameters that control the statistical models assumed by SHyGAMP, one based on the expectation-maximization (EM) methodology from [4] and the other based on a variation of the

Stein's unbiased risk estimate (SURE) methodology from [35]. We also give numerical evidence that these methods yield near-optimal hyperparameter estimates.

Finally, in Section 3.5, we compare our proposed SHyGAMP methods to the state-of-the-art MLR approaches [26, 27] on both synthetic and practical real-world problems. Our experiments suggest that our proposed methods offer simultaneous improvements in classification error rate and runtime.

Notation: Random quantities are typeset in sans-serif (e.g., x) while deterministic quantities are typeset in serif (e.g., x). The pdf of random variable x under deterministic parameters $\boldsymbol{\theta}$ is written as $p_{\mathsf{x}}(x; \boldsymbol{\theta})$, where the subscript and parameterization are sometimes omitted for brevity. Column vectors are typeset in boldface lower-case (e.g., $\boldsymbol{y}$ or $\mathbf{\mathsf{y}}$), matrices in boldface upper-case (e.g., $\boldsymbol{X}$ or $\mathbf{\mathsf{X}}$), and their transpose is denoted by $(\cdot)^{\mathsf{T}}$. $\mathrm{E}\{\cdot\}$ denotes expectation and $\mathrm{Cov}\{\cdot\}$ autocovariance. $\boldsymbol{I}_K$ denotes the $K \times K$ identity matrix, $\boldsymbol{e}_k$ the kth column of $\boldsymbol{I}_K$, $\mathbf{1}_K$ the length-K vector of ones, and $\mathrm{diag}(\boldsymbol{b})$ the diagonal matrix created from the vector $\boldsymbol{b}$. $[\boldsymbol{B}]_{m,n}$ denotes the element in the m^{th} row and n^{th} column of $\boldsymbol{B}$, and $\|\cdot\|_F$ the Frobenius norm. Finally, δ_n denotes the Kronecker delta sequence, $\delta(x)$ the Dirac delta distribution, and 1_A the indicator function of the event A.

3.2 HyGAMP for Multiclass Classification

In this section, we detail the application of HyGAMP [13] from Chapter 2 to multiclass linear classification. In particular, we show that the sum-product algorithm (SPA) variant of HyGAMP is a loopy belief propagation (LBP) approximation of the classification-error-rate minimizing linear classifier and that the min-sum algorithm (MSA) variant is an LBP approach to solving the MAP problem (3.10).

3.2.1 Classification via sum-product HyGAMP

Suppose that we are given M labeled training pairs $\{(y_m, \boldsymbol{a}_m)\}_{m=1}^{M}$ and T test feature vectors $\{\boldsymbol{a}_t\}_{t=M+1}^{M+T}$ associated with unknown test labels $\{\mathsf{y}_t\}_{t=M+1}^{M+T}$, all obeying the MLR statistical model (3.2)-(3.4). Consider the problem of computing the classification-error-rate minimizing hypotheses $\{\widehat{y}_t\}_{t=M+1}^{M+T}$,

$$\widehat{y}_t = \arg \max_{y_t \in \{1,\ldots,D\}} p_{\mathsf{y}_t|\mathbf{y}_{1:M}}\Big(y_t \,\Big|\, \boldsymbol{y}_{1:M}; \boldsymbol{A}\Big), \tag{3.11}$$

under known $p_{\mathsf{y}|\mathbf{z}}$ and $p_{\mathbf{x}}$, where $\boldsymbol{y}_{1:M} \triangleq [y_1, \ldots, y_M]^{\mathsf{T}}$ and $\boldsymbol{A} \triangleq [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{M+T}]^{\mathsf{T}}$. The probabilities in (3.11) can be computed via the marginalization

$$p_{\mathsf{y}_t|\mathbf{y}_{1:M}}\Big(y_t \,\Big|\, \boldsymbol{y}_{1:M}; \boldsymbol{A}\Big) = p_{\mathsf{y}_t,\mathbf{y}_{1:M}}\Big(y_t, \boldsymbol{y}_{1:M}; \boldsymbol{A}\Big) Z_{\mathbf{y}}^{-1} \tag{3.12}$$

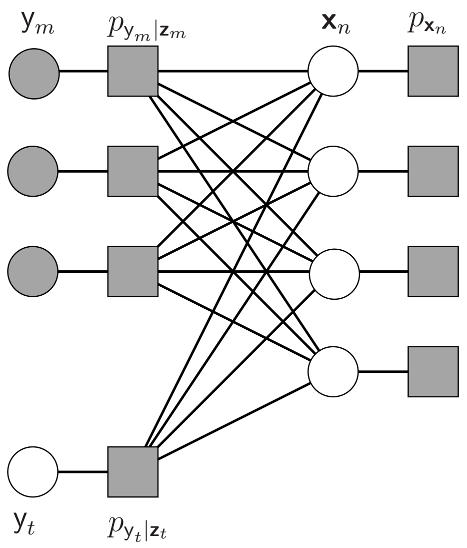
$$= Z_{\mathbf{y}}^{-1} \sum_{\boldsymbol{y} \in \mathcal{Y}_t(y_t)} \int p_{\mathbf{y},\mathbf{X}}(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{A})\, \mathrm{d}\boldsymbol{X}, \tag{3.13}$$

with scaling constant $Z_{\mathbf{y}}^{-1}$, label vector $\boldsymbol{y} = [y_1, \ldots, y_{M+T}]^{\mathsf{T}}$, and constraint set $\mathcal{Y}_t(y) \triangleq \Big\{\widetilde{\boldsymbol{y}} \in \{1, \ldots, K\}^{M+T} \text{ s.t. } [\widetilde{\boldsymbol{y}}]_t = y \text{ and } [\widetilde{\boldsymbol{y}}]_m = y_m \ \forall m = 1, \ldots, M\Big\}$, which fixes the tth element of $\boldsymbol{y}$ at the value y and the first M elements of $\boldsymbol{y}$ at the values of the corresponding training labels. Due to (3.2) and (3.4), the joint pdf in (3.13) factors as

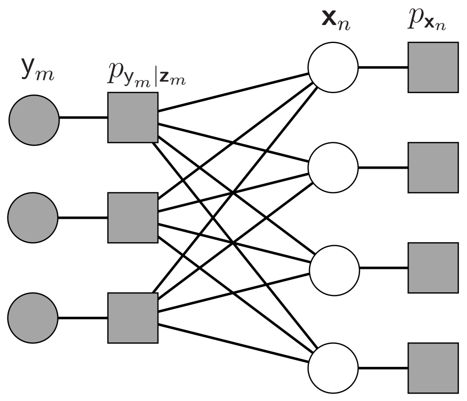
$$p_{\mathbf{y},\mathbf{X}}(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{A}) \quad = \quad \prod_{m=1}^{M+T} p_{\mathsf{y}|\mathbf{z}}(y_m \,|\, \boldsymbol{X}^{\mathsf{T}} \boldsymbol{a}_m) \prod_{n=1}^{N} p_{\mathbf{x}}(\boldsymbol{x}_n). \tag{3.14}$$

The factorization in (3.14) is depicted by the *factor graph* in Figure 3.1a, where the random variables $\{\mathsf{y}_m\}$ and random vectors $\{\mathbf{x}_n\}$ are connected to the pdf factors in which they appear.

Since exact computation of the marginal posterior test-label probabilities is an NP-hard problem [36], we are interested in alternative strategies, such as those based on loopy belief propagation by the SPA [31]. Although a direct application of the SPA


y_m
$p_{\mathsf{y}_m|\mathbf{z}_m}$
$\mathbf{x}_n$
$p_{\mathbf{x}_n}$
y_t
$p_{\mathsf{y}_t|\mathbf{z}_t}$

(a) Full



(b) Reduced

Figure 3.1: Factor graph representations of (3.14), with white/gray circles denoting unobserved/observed random variables, and gray rectangles denoting pdf "factors".

is itself intractable when $p_{\mathsf{y}|\mathsf{z}}$ takes the MLR form (3.3), the SPA simplifies in the large-system limit under i.i.d. sub-Gaussian $\boldsymbol{A}$, leading to the HyGAMP approximation [13] given[4] in Table 2.1. Although in practical MLR applications $\boldsymbol{A}$ is not i.i.d. Gaussian,[5] the numerical results in Section 3.5 suggest that treating it as such works sufficiently well.

We note from Figure 3.1a that the HyGAMP algorithm is applicable to a factor graph with vector-valued variable nodes. As such, it generalizes the GAMP algorithm from [2], which applies only to a factor graph with scalar-variable nodes. Below, we give a brief explanation for the steps in Table 2.1. For those interested in more details, we suggest [13] for an overview and derivation of HyGAMP, [2] for an overview and derivation of GAMP, [37] for rigorous analysis of GAMP under large i.i.d. sub-Gaussian $\boldsymbol{A}$, and [38, 39] for fixed-point and local-convergence analysis of GAMP under arbitrary $\boldsymbol{A}$.

Lines 19-20 of Table 2.1 produce an approximation of the posterior mean and covariance of x_n at each iteration t. Similarly, lines 8-9 produce an approximation of the posterior mean and covariance of $\mathsf{z}_m \triangleq \mathsf{X}^{\mathsf{T}}\boldsymbol{a}_m$. The posterior mean and covariance of x_n are computed from the intermediate quantity $\widehat{\boldsymbol{r}}_n(t)$, which behaves like a noisy measurement of the true $\boldsymbol{x}_n$. In particular, for i.i.d. Gaussian $\boldsymbol{A}$ in the large-system limit, $\widehat{\boldsymbol{r}}_n(t)$ is a typical realization of the random vector $\mathsf{r}_n = \boldsymbol{x}_n + \mathsf{v}_n$ with $\mathsf{v}_n \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}^{\mathsf{r}}_n(t))$. Thus, the approximate posterior pdf used in lines 19-20 is

$$p_{\mathsf{x}|\mathsf{r}}(\boldsymbol{x}_n|\widehat{\boldsymbol{r}}_n; \boldsymbol{Q}^{\mathsf{r}}_n) = \frac{p_{\mathsf{x}}(\boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}^{\mathsf{r}}_n)}{\int p_{\mathsf{x}}(\boldsymbol{x}'_n)\mathcal{N}(\boldsymbol{x}'_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}^{\mathsf{r}}_n)\,\mathrm{d}\boldsymbol{x}'_n}. \tag{3.15}$$

[4] The HyGAMP algorithm in [13] is actually more general than what is specified in Table 2.1, but the version in Table 2.1 is sufficient to handle the factor graph in Figure 3.1a.

[5] We note that many of the standard data pre-processing techniques, such as z-scoring, tend to make the feature distributions closer to zero-mean Gaussian.

A similar interpretation holds for HyGAMP's approximation of the posterior mean and covariance of z_m in lines 8-9, which uses the intermediate vector $\widehat{\boldsymbol{p}}_m(t)$ and the approximate posterior pdf

$$\begin{aligned} & p_{\mathsf{z}|\mathsf{y},\mathsf{p}}(\boldsymbol{z}_m|y_m,\widehat{\boldsymbol{p}}_m;\boldsymbol{Q}^{\mathsf{p}}_m) \\ &= \frac{p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}_m)}{\int p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}'_m)\mathcal{N}(\boldsymbol{z}'_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}_m)\,\mathrm{d}\boldsymbol{z}'_m}. \end{aligned} \tag{3.16}$$

3.2.2 Classification via min-sum HyGAMP

As discussed in Section 3.1.2, an alternative approach to linear classification and feature selection is through MAP estimation of the true weight matrix $\boldsymbol{X}$. Given a likelihood of the form (3.2) and a prior of the form (3.4), the MAP estimate is the solution to the optimization problem (3.10).

Similar to how the SPA can be used to compute approximate marginal posteriors in loopy graphs, the min-sum algorithm (MSA) [31] can be used to compute the MAP estimate. Although a direct application of the MSA is intractable when $p_{\mathsf{y}|\mathsf{z}}$ takes the MLR form (3.3), the MSA simplifies in the large-system limit under i.i.d. sub-Gaussian $\boldsymbol{A}$, leading to the `MSA` form of HyGAMP specified in Table 2.1.

As described in Section 3.2.1, when $\boldsymbol{A}$ is large and i.i.d. sub-Gaussian, the vector $\widehat{\boldsymbol{r}}_n(t)$ in Table 2.1 behaves like a Gaussian-noise-corrupted observation of the true $\boldsymbol{x}_n$ with noise covariance $\boldsymbol{Q}^{\mathsf{r}}_n(t)$. Thus, line 16 can be interpreted as MAP estimation of $\boldsymbol{x}_n$ and line 17 as measuring the local curvature of the corresponding MAP cost. Similar interpretations hold for MAP estimation of $\boldsymbol{z}_m$ via lines 5-6.

3.2.3 Implementation of sum-product HyGAMP

From Table 2.1, we see that HyGAMP requires inverting $M+N$ matrices of size $K \times K$ (for lines 11 and 13) in addition to solving $M+N$ joint inference problems of dimension K in lines 16-20 and 5-9. We now briefly discuss the latter problems for the sum-product version of HyGAMP.

Inference of $\boldsymbol{x}_n$

One choice of weight-coefficient prior p_{x_n} that facilitates row-sparse $\boldsymbol{X}$ and tractable SPA inference is Bernoulli-multivariate-Gaussian, i.e.,

$$p_{\mathsf{x}}(\boldsymbol{x}_n) = (1-\beta)\delta(\boldsymbol{x}_n) + \beta\mathcal{N}(\boldsymbol{x}_n; \boldsymbol{0}, v\boldsymbol{I}), \tag{3.17}$$

where $\delta(\cdot)$ denotes the Dirac delta and $\beta \in (0,1]$. In this case, it can be shown [14] that the mean and variance computations in lines 19-20 of Table 2.1 reduce to

$$C_n = 1 + \frac{1-\beta}{\beta}\frac{\mathcal{N}(\boldsymbol{0}; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}^{\mathbf{r}}_n)}{\mathcal{N}(\boldsymbol{0}; \widehat{\boldsymbol{r}}_n, v\boldsymbol{I} + \boldsymbol{Q}^{\mathbf{r}}_n)} \tag{3.18}$$

$$\widehat{\boldsymbol{x}}_n = C_n^{-1}(\boldsymbol{I} + v^{-1}\boldsymbol{Q}^{\mathbf{r}}_n)^{-1}\widehat{\boldsymbol{r}}_n \tag{3.19}$$

$$\boldsymbol{Q}^{\mathbf{x}}_n = C_n^{-1}(\boldsymbol{I} + v^{-1}\boldsymbol{Q}^{\mathbf{r}}_n)^{-1}\boldsymbol{Q}^{\mathbf{r}}_n + (C_n - 1)\widehat{\boldsymbol{x}}_n\widehat{\boldsymbol{x}}_n^{\mathsf{T}}, \tag{3.20}$$

which requires a $K \times K$ matrix inversion at each n.

Inference of $\boldsymbol{z}_m$

When $p_{\mathsf{y}|\mathbf{z}}$ takes the MLR form in (3.3), closed-form expressions for $\widehat{\boldsymbol{z}}_m(t)$ and $\boldsymbol{Q}^{\mathbf{z}}_m(t)$ from lines 8-9 of Table 2.1 do not exist. While these computations could be approximated using, e.g., numerical integration or importance sampling, this is expensive because $\widehat{\boldsymbol{z}}_m(t)$ and $\boldsymbol{Q}^{\mathbf{z}}_m(t)$ must be computed for every index m at every HyGAMP iteration t. More details on these approaches will be presented in Section 3.3.2, in the context of SHyGAMP.

3.2.4 Implementation of min-sum HyGAMP

Inference of $\boldsymbol{x}_n$

To ease the computation of line 16 in Table 2.1, it is typical to choose a log-concave prior p_{x} so that the optimization problem (3.10) is concave (since $p_{\mathsf{y}|\mathsf{z}}$ in (3.3) is also log-concave). As discussed in Section 3.1.2, a common example of a log-concave sparsity-promoting prior is the Laplace prior (3.5). In this case, line 16 becomes

$$\widehat{\boldsymbol{x}}_n = \arg\max_{\boldsymbol{x}} -\frac{1}{2}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}}[\boldsymbol{Q}^{\mathsf{r}}_n]^{-1}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n) - \lambda\|\boldsymbol{x}\|_1, \tag{3.21}$$

which is essentially the LASSO [40] problem. Although (3.21) has no closed-form solution, it can be solved iteratively using, e.g., minorization-maximization (MM) [41].

To maximize a function $J(\boldsymbol{x})$, MM iterates the recursion

$$\widehat{\boldsymbol{x}}^{(t+1)} = \arg\max_{\boldsymbol{x}} \widehat{J}(\boldsymbol{x}; \widehat{\boldsymbol{x}}^{(t)}), \tag{3.22}$$

where $\widehat{J}(\boldsymbol{x}; \widehat{\boldsymbol{x}})$ is a surrogate function that minorizes $J(\boldsymbol{x})$ at $\widehat{\boldsymbol{x}}$. In other words, $\widehat{J}(\boldsymbol{x}; \widehat{\boldsymbol{x}}) \leq J(\widehat{\boldsymbol{x}})\ \forall \boldsymbol{x}$ for any fixed $\widehat{\boldsymbol{x}}$, with equality when $\boldsymbol{x} = \widehat{\boldsymbol{x}}$. To apply MM to (3.21), we identify the utility function as $J_n(\boldsymbol{x}) \triangleq -\frac{1}{2}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}}[\boldsymbol{Q}^{\mathsf{r}}_n]^{-1}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n) - \lambda\|\boldsymbol{x}\|_1$. Next we apply a result from [42] that established that $J_n(\boldsymbol{x})$ is minorized by $\widehat{J}_n(\boldsymbol{x}; \widehat{\boldsymbol{x}}_n^{(t)}) \triangleq -\frac{1}{2}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}}[\boldsymbol{Q}^{\mathsf{r}}_n]^{-1}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n) - \frac{\lambda}{2}\Big(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}_n^{(t)})\boldsymbol{x} + \|\widehat{\boldsymbol{x}}_n^{(t)}\|_2^2\Big)$ with $\boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}) \triangleq \operatorname{diag}\Big\{|\widehat{x}_1|^{-1}, \ldots, |\widehat{x}_D|^{-1}\Big\}$. Thus (3.22) implies

$$\begin{aligned}
\widehat{\boldsymbol{x}}_n^{(t+1)} &= \arg\max_{\boldsymbol{x}} \widehat{J}_n(\boldsymbol{x}; \widehat{\boldsymbol{x}}_n^{(t)}) && (3.23)\\
&= \arg\max_{\boldsymbol{x}} \boldsymbol{x}^{\mathsf{T}}[\boldsymbol{Q}^{\mathsf{r}}_n]^{-1}\widehat{\boldsymbol{r}}_n - \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\Big([\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} + \lambda\boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}_n^{(t)})\Big)\boldsymbol{x} && (3.24)\\
&= \Big([\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} + \lambda\boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}_n^{(t)})\Big)^{-1}[\boldsymbol{Q}^{\mathsf{r}}_n]^{-1}\widehat{\boldsymbol{r}}_n && (3.25)
\end{aligned}$$

where (3.24) dropped the $\boldsymbol{x}$-invariant terms from $\widehat{J}_n(\boldsymbol{x}; \widehat{\boldsymbol{x}}_n^{(t)})$. Note that each iteration t of (3.25) requires a $K \times K$ matrix inverse for each n.

Line 17 of Table 2.1 then says to set $\boldsymbol{Q}_n^{\mathsf{x}}$ equal to the Hessian of the objective function in (3.21) at $\widehat{\boldsymbol{x}}_n$. Recalling that the second derivative of $|x_{nk}|$ is undefined when $x_{nk} = 0$ but otherwise equals zero, we set $\boldsymbol{Q}_n^{\mathsf{x}} = \boldsymbol{Q}_n^{\mathsf{r}}$ but then zero the kth row and column of $\boldsymbol{Q}_n^{\mathsf{x}}$ for all k such that $\widehat{x}_{nk} = 0$.

Inference of $\boldsymbol{z}_m$

Min-sum HyGAMP also requires the computation of lines 5-6 in Table 2.1. In our MLR application, line 5 reduces to the concave optimization problem

$$\begin{aligned}\widehat{\boldsymbol{z}}_m = \arg\max_{\boldsymbol{z}} & -\frac{1}{2}(\boldsymbol{z} - \widehat{\boldsymbol{p}}_m)^{\mathsf{T}}[\boldsymbol{Q}_m^{\mathsf{p}}]^{-1}(\boldsymbol{z} - \widehat{\boldsymbol{p}}_m) \\ & + \log p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}). \end{aligned} \tag{3.26}$$

Although (3.26) can be solved in a variety of ways (see [14] for MM-based methods), we now describe one based on Newton's method [43], i.e.,

$$\widehat{\boldsymbol{z}}_m^{(t+1)} = \widehat{\boldsymbol{z}}_m^{(t)} - \alpha^{(t)}[\boldsymbol{H}_m^{(t)}]^{-1}\boldsymbol{g}_m^{(t)}, \tag{3.27}$$

where $\boldsymbol{g}_m^{(t)}$ and $\boldsymbol{H}_m^{(t)}$ are the gradient and Hessian of the objective function in (3.26) at $\widehat{\boldsymbol{z}}_m^{(t)}$, and $\alpha^{(t)} \in (0, 1]$ is a stepsize. From (3.3), it can be seen that $\frac{\partial}{\partial z_i}\log p_{\mathsf{y}|\mathbf{z}}(y|\boldsymbol{z}) = \delta_{y-i} - p_{\mathsf{y}|\mathbf{z}}(i|\boldsymbol{z})$, and so

$$\boldsymbol{g}_m^{(t)} = \boldsymbol{u}(\widehat{\boldsymbol{z}}_m^{(t)}) - \boldsymbol{e}_{y_m} + [\boldsymbol{Q}_m^{\mathsf{p}}]^{-1}(\widehat{\boldsymbol{z}}_m^{(t)} - \widehat{\boldsymbol{p}}_m), \tag{3.28}$$

where $\boldsymbol{e}_y$ denotes the yth column of $\boldsymbol{I}_K$ and $\boldsymbol{u}(\boldsymbol{z}) \in \mathbb{R}^{K\times 1}$ is defined elementwise as

$$[\boldsymbol{u}(\boldsymbol{z})]_i \triangleq p_{\mathsf{y}|\mathbf{z}}(i|\boldsymbol{z}). \tag{3.29}$$

Similarly, it is known [44] that the Hessian takes the form

$$\boldsymbol{H}_m^{(t)} = \boldsymbol{u}(\widehat{\boldsymbol{z}}_m^{(t)})\boldsymbol{u}(\widehat{\boldsymbol{z}}_m^{(t)})^{\mathsf{T}} - \operatorname{diag}\{\boldsymbol{u}(\widehat{\boldsymbol{z}}_m^{(t)})\} - [\boldsymbol{Q}_m^{\mathsf{p}}]^{-1}, \tag{3.30}$$

which also provides the answer to line 6 of Table 2.1. Note that each iteration t of (3.27) requires a $K \times K$ matrix inverse for each m.

It is possible to circumvent the matrix inversion in (3.27) via componentwise update, i.e.,

$$\widehat{z}_{mk}^{(t+1)} = \widehat{z}_{mk}^{(t)} - \alpha^{(t)} g_{mk}^{(t)} / H_{mk}^{(t)}, \tag{3.31}$$

where $g_{mk}^{(t)}$ and $H_{mk}^{(t)}$ are the first and second derivatives of the objective function in (3.26) with respect to z_k at $\boldsymbol{z} = \widehat{\boldsymbol{z}}_m^{(t)}$. From (3.28)-(3.30), it follows that

$$g_{mk}^{(t)} = p_{\mathsf{y}|\mathbf{z}}(k|\widehat{\boldsymbol{z}}_m^{(t)}) - \delta_{y_m - k} + \Big[[\boldsymbol{Q}_m^{\mathsf{p}}]^{-1}\Big]_{:,k}^{\mathsf{T}} (\widehat{\boldsymbol{z}}_m^{(t)} - \widehat{\boldsymbol{p}}_m) \tag{3.32}$$

$$H_{mk}^{(t)} = p_{\mathsf{y}|\mathbf{z}}(k|\widehat{\boldsymbol{z}}_m^{(t)})^2 - p_{\mathsf{y}|\mathbf{z}}(k|\widehat{\boldsymbol{z}}_m^{(t)}) - \Big[[\boldsymbol{Q}_m^{\mathsf{p}}]^{-1}\Big]_{kk}. \tag{3.33}$$

3.2.5 HyGAMP summary

In summary, the SPA and MSA variants of the HyGAMP algorithm provide tractable methods of approximating the posterior test-label probabilities $p_{\mathsf{y}_t|\mathbf{y}_{1:M}}\Big(y_t \,\Big|\, \boldsymbol{y}_{1:M}; \boldsymbol{A}\Big)$ and computing the MAP weight matrix $\widehat{\boldsymbol{X}} = \arg\max_{\boldsymbol{X}} p_{\mathbf{y}_{1:M},\mathbf{X}}(\boldsymbol{y}_{1:M}, \boldsymbol{X}; \boldsymbol{A})$ respectively, under a separable likelihood (3.2) and a separable prior (3.4). In particular, HyGAMP attacks the high-dimensional inference problems of interest using a sequence of $M + N$ low-dimensional (in particular, K-dimensional) inference problems and $K \times K$ matrix inversions, as detailed in Table 2.1.

As detailed in the previous subsections, however, these K-dimensional inference problems are non-trivial in the sparse MLR case, making HyGAMP computationally costly. We refer the reader to Table 3.1 for a summary of the K-dimensional inference problems encountered in running SPA-HyGAMP or MSA-HyGAMP, as well as their

Algorithm	Quantity	Method	Complexity
SPA-HyGAMP	$\widehat{\boldsymbol{x}}$	CF	$O(D^3)$
	$\boldsymbol{Q}^{\mathsf{x}}$	CF	$O(K^3)$
	$\widehat{\boldsymbol{z}}$	NI	$O(K^T)$
	$\boldsymbol{Q}^{\mathsf{z}}$	NI	$O(KD^T)$
MSA-HyGAMP	$\widehat{\boldsymbol{x}}$	MM	$O(TK^3)$
	$\boldsymbol{Q}^{\mathsf{x}}$	CF	$O(K^3)$
	$\widehat{\boldsymbol{z}}$	CWN	$O(TK^2+K^3)$
	$\boldsymbol{Q}^{\mathsf{z}}$	CF	$O(K^3)$

Table 3.1: A summary of the D-dimensional inference sub-problems encountered when running SPA-HyGAMP or MSA-HyGAMP, as well as their associated computational costs. 'CF' = 'closed form', 'NI' = 'numerical integration', 'MM' = 'minorization-maximization', and 'CWN' = 'component-wise Newton's method'. For the NI method, T denotes the number of samples per dimension, and for the MM and CWN methods T denotes the number of iterations.

associated computational costs. Thus, in the sequel, we propose a computationally efficient simplification of HyGAMP that, as we will see in Section 3.5, compares favorably with existing state-of-the-art methods.

3.3 SHyGAMP for Multiclass Classification

As described in Section 3.2, a direct application of HyGAMP to sparse MLR is computationally costly. Thus, in this section, we propose a *simplified HyGAMP* (SHyGAMP) algorithm for sparse MLR, whose complexity is greatly reduced. The simplification itself is rather straightforward: we constrain the covariance matrices $\boldsymbol{Q}_n^{\mathsf{r}}$, $\boldsymbol{Q}_n^{\mathsf{x}}$, $\boldsymbol{Q}_m^{\mathsf{p}}$, and $\boldsymbol{Q}_m^{\mathsf{z}}$ to be diagonal. In other words,

$$\boldsymbol{Q}_n^{\mathsf{r}} = \text{diag}\left\{q_{n1}^{\mathsf{r}}, \ldots, q_{nK}^{\mathsf{r}}\right\}, \tag{3.34}$$

and similar for $\boldsymbol{Q}_n^{\mathsf{x}}$, $\boldsymbol{Q}_m^{\mathsf{p}}$, and $\boldsymbol{Q}_m^{\mathsf{z}}$. As a consequence, the $K \times K$ matrix inversions in lines 11 and 13 of Table 2.1 each reduce to K scalar inversions. More importantly, the

K-dimensional inference problems in lines 16-20 and 5-9 can be tackled using much simpler methods than those described in Section 3.2, as we detail below.

We further approximate the SHyGAMP algorithm using the *scalar variance* GAMP approximation from [13], which reduces the memory and complexity of the algorithm, which we described in detail in Section 2.4.

3.3.1 Sum-product SHyGAMP: Inference of $\boldsymbol{x}_n$

With diagonal $\boldsymbol{Q}^{\mathsf{r}}_n$ and $\boldsymbol{Q}^{\mathsf{x}}_n$, the implementation of lines 19-20 is greatly simplified by choosing a sparsifying prior $p_{\mathbf{x}}$ with the separable form $p_{\mathbf{x}}(\boldsymbol{x}_n) = \prod_{k=1}^{K} p_{\mathsf{x}}(x_{nk})$. A common example is the Bernoulli-Gaussian (BG) prior

$$p_{\mathsf{x}}(x_{nk}) = (1 - \beta_k)\delta(x_{nk}) + \beta_d \mathcal{N}(x_{nk}; m_k, v_k \boldsymbol{I}). \tag{3.35}$$

For any separable $p_{\mathbf{x}}$, lines 19-20 reduce to computing the mean and variance of the distribution

$$p_{\mathsf{x}|\mathsf{r}}(x_{nk}|\widehat{r}_{nk}; q^{\mathsf{r}}_{nk}) = \frac{p_{\mathsf{x}}(x_{nk})\mathcal{N}(x_{nk}; \widehat{r}_{nk}, q^{\mathsf{r}}_{nk})}{\int p_{\mathsf{x}}(x'_{nk})\mathcal{N}(x'_{nk}; \widehat{r}_{nk}, q^{\mathsf{r}}_{nk})\,\mathrm{d}x'_{nk}}. \tag{3.36}$$

for all $n = 1 \dots N$ and $k = 1 \dots K$, as in the simpler GAMP algorithm [2]. With the BG prior (3.35), these quantities can be computed in closed form (see, e.g., [45]).

3.3.2 Sum-product SHyGAMP: Inference of $\boldsymbol{z}_m$

With diagonal $\boldsymbol{Q}^{\mathbf{p}}_m$ and $\boldsymbol{Q}^{\mathbf{z}}_m$, the implementation of lines 8-9 can also be greatly simplified. Essentially, the problem becomes that of computing the scalar means and variances

$$\widehat{z}_{mk} = C_m^{-1} \int_{\mathbb{R}^K} z_k \, p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}) \prod_{d=1}^{K} \mathcal{N}(z_d; \widehat{p}_{md}, q^{\mathbf{p}}_{md})\,\mathrm{d}\boldsymbol{z} \tag{3.37}$$

$$q^{\mathbf{z}}_{mk} = C_m^{-1} \int_{\mathbb{R}^K} z_k^2 \, p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}) \prod_{d=1}^{K} \mathcal{N}(z_d; \widehat{p}_{md}, q^{\mathbf{p}}_{md})\,\mathrm{d}\boldsymbol{z} - \widehat{z}_{mk}^2 \tag{3.38}$$

for $m = 1 \dots M$ and $k = 1 \dots K$. Here, $p_{\mathsf{y}|\mathsf{z}}$ has the MLR form in (3.3) and C_m is a normalizing constant defined as

$$C_m \triangleq \int_{\mathbb{R}^K} p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}) \prod_{d=1}^{K} \mathcal{N}(z_d; \widehat{p}_{md}, q^{\mathsf{p}}_{md}) \, \mathrm{d}\boldsymbol{z}. \tag{3.39}$$

Note that the likelihood $p_{\mathsf{y}|\mathsf{z}}$ is not separable and so inference does not decouple across k, as it did in (3.36). We now describe several approaches to computing (3.37)-(3.38).

Numerical integration

A straightforward approach to (approximately) computing (3.37)-(3.39) is through numerical integration (NI). For this, we propose to use a hyper-rectangular grid of $\boldsymbol{z}$ values where, for z_k, the interval $\left[\widehat{p}_{mk} - \alpha\sqrt{q^{\mathsf{p}}_{mk}},\ \widehat{p}_{mk} + \alpha\sqrt{q^{\mathsf{p}}_{mk}}\right]$ is sampled at T equi-spaced points. Because a K-dimensional numerical integral must be computed for each index m and k, the complexity of this approach grows as $O(MKT^K)$, making it impractical unless K, the number of classes, is very small.

Importance sampling

An alternative approximation of (3.37)-(3.39) can be obtained through importance sampling (IS) [23, §11.1.4]. Here, we draw T independent samples $\{\widetilde{\boldsymbol{z}}_m[t]\}_{t=1}^{T}$ from $\mathcal{N}(\widehat{\boldsymbol{p}}_m, \boldsymbol{Q}^{\mathsf{p}}_m)$ and compute

$$C_m \approx \sum_{t=1}^{T} p_{\mathsf{y}|\mathsf{z}}(y_m|\widetilde{\boldsymbol{z}}_m[t]) \tag{3.40}$$

$$\widehat{z}_{mk} \approx C_m^{-1} \sum_{t=1}^{T} \widetilde{z}_{mk}[t] p_{\mathsf{y}|\mathsf{z}}(y_m|\widetilde{\boldsymbol{z}}_m[t]) \tag{3.41}$$

$$q^{\mathsf{z}}_{mk} \approx C_m^{-1} \sum_{t=1}^{T} \widetilde{z}^2_{mk}[t] p_{\mathsf{y}|\mathsf{z}}(y_m|\widetilde{\boldsymbol{z}}_m[t]) - \widehat{z}^2_{mk} \tag{3.42}$$

for all m and k. The complexity of this approach grows as $O(MKT)$.

Taylor-series approximation

Another approach is to approximate the likelihood $p_{\mathsf{y}|\mathbf{z}}$ using a second-order Taylor series (TS) about $\widehat{\boldsymbol{p}}_m$, i.e., $p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}) \approx f_m(\boldsymbol{z};\widehat{\boldsymbol{p}}_m)$ with

$$\begin{aligned} f_m(\boldsymbol{z};\widehat{\boldsymbol{p}}_m) &\triangleq p_{\mathsf{y}|\mathbf{z}}(y_m|\widehat{\boldsymbol{p}}_m) + \boldsymbol{g}_m(\widehat{\boldsymbol{p}}_m)^{\mathsf{T}}(\boldsymbol{z}-\widehat{\boldsymbol{p}}_m) \\ &\quad + \frac{1}{2}(\boldsymbol{z}-\widehat{\boldsymbol{p}}_m)^{\mathsf{T}}\boldsymbol{H}_m(\widehat{\boldsymbol{p}}_m)(\boldsymbol{z}-\widehat{\boldsymbol{p}}_m) \end{aligned} \tag{3.43}$$

for gradient $\boldsymbol{g}_m(\widehat{\boldsymbol{p}}) \triangleq \frac{\partial}{\partial \boldsymbol{z}} p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z})\Big|_{\boldsymbol{z}=\widehat{\boldsymbol{p}}}$ and Hessian $\boldsymbol{H}_m(\widehat{\boldsymbol{p}}) \triangleq \frac{\partial^2}{\partial \boldsymbol{z}^2} p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z})\Big|_{\boldsymbol{z}=\widehat{\boldsymbol{p}}}$. In this case, it can be shown [14] that

$$C_m \approx f_m(\widehat{\boldsymbol{p}}_m) + \frac{1}{2}\sum_{k=1}^{K} H_{mk}(\widehat{\boldsymbol{p}}_m) q^{\mathbf{p}}_{mk} \tag{3.44}$$

$$\begin{aligned} \widehat{z}_{md} \approx \widehat{C}_m^{-1}\Bigg(& f_m(\widehat{\boldsymbol{p}}_m)\,\widehat{p}_{mk} + g_{mk}(\widehat{\boldsymbol{p}}_m) q^{\mathbf{p}}_{mk} \\ & + \frac{1}{2}\sum_{k=1}^{K} \widehat{p}_{mk} q^{\mathbf{p}}_{mk} H_{mk}(\widehat{\boldsymbol{p}}_m)\Bigg) \end{aligned} \tag{3.45}$$

$$\begin{aligned} q^{\mathbf{z}}_{mk} \approx C_m^{-1}\Bigg(& f_m(\widehat{\boldsymbol{p}}_m)\,(\widehat{p}^2_{mk} + q^{\mathbf{p}}_{mk}) + 2 g_{mk}(\widehat{\boldsymbol{p}}_m)\widehat{p}_{mk} q^{\mathbf{p}}_{mk} \\ & + \frac{1}{2} q^{\mathbf{p}}_{mk}\Big(\widehat{p}^2_{mk} + 3 q^{\mathbf{p}}_{mk}\Big) H_{mk}(\widehat{\boldsymbol{p}}_m) \\ & + \frac{1}{2}\Big(\widehat{p}^2_{mk} + q^{\mathbf{p}}_{mk}\Big) H_{mk}(\widehat{\boldsymbol{p}}_m) \sum_{k' \neq k} q^{\mathbf{p}}_{mk'}\Bigg) - \widehat{z}^2_{mk}, \end{aligned} \tag{3.46}$$

where $H_{mk}(\widehat{\boldsymbol{p}}) \triangleq [\boldsymbol{H}_m(\widehat{\boldsymbol{p}})]_{kk}$. The complexity of this approach grows as $O(MK)$.

Gaussian mixture approximation

It is known that the logistic cdf $1/(1+\exp(-x))$ is well approximated by a mixture of a few Gaussian cdfs, which leads to an efficient method of approximating (3.37)-(3.38) in the case of *binary* logistic regression (i.e., $K = 2$) [46]. We now develop an extension of this method for the MLR case (i.e., $K \geq 2$).

To facilitate the Gaussian mixture (GM) approximation, we work with the difference variables

$$\gamma_k^{(y)} \triangleq \begin{cases} z_y - z_k & k \neq y \\ z_y & k = y \end{cases}. \tag{3.47}$$

Their utility can be seen from the fact that (recalling (3.3))

$$\begin{aligned} p_{\mathsf{y}|\mathbf{z}}(y|\boldsymbol{z}) &= \frac{1}{1 + \sum_{k\neq y} \exp(z_k - z_y)} & (3.48) \\ &= \frac{1}{1 + \sum_{k\neq y} \exp(-\gamma_k^{(y)})} \triangleq l^{(y)}(\boldsymbol{\gamma}^{(y)}), & (3.49) \end{aligned}$$

which is smooth, positive, and bounded by 1, and strictly increasing in $\gamma_k^{(y)}$. Thus,[6] for appropriately chosen $\{\alpha_l, \mu_{kl}, \sigma_{kl}\}$,

$$l^{(y)}(\boldsymbol{\gamma}) \approx \sum_{l=1}^{L} \alpha_l \prod_{k\neq y} \Phi\left(\frac{\gamma_k - \mu_{kl}}{\sigma_{kl}}\right) \triangleq \widehat{l}^{(y)}(\boldsymbol{\gamma}), \tag{3.50}$$

where $\Phi(x)$ is the standard normal cdf, $\sigma_{kl} > 0$, $\alpha_l \geq 0$, and $\sum_l \alpha_l = 1$. In practice, the GM parameters $\{\alpha_l, \mu_{kl}, \sigma_{kl}\}$ could be designed off-line to minimize, e.g., the total variation distance $\sup_{\boldsymbol{\gamma}\in\mathbb{R}^K} |l^{(y)}(\boldsymbol{\gamma}) - \widehat{l}^{(y)}(\boldsymbol{\gamma})|$.

Recall from (3.37)-(3.39) that our objective is to compute quantities of the form

$$\int_{\mathbb{R}^K} (\boldsymbol{e}_k^{\mathsf{T}}\boldsymbol{z})^i \, p_{\mathsf{y}|\mathbf{z}}(y|\boldsymbol{z}) \mathcal{N}(\boldsymbol{z}; \widehat{\boldsymbol{p}}, \boldsymbol{Q}^{\mathsf{p}}) \, \mathrm{d}\boldsymbol{z} \triangleq S_{ki}^{(y)}, \tag{3.51}$$

where $i \in \{0, 1, 2\}$, $\boldsymbol{Q}^{\mathsf{p}}$ is diagonal, and $\boldsymbol{e}_k$ is the kth column of $\boldsymbol{I}_K$. To exploit (3.50), we change the integration variable to

$$\boldsymbol{\gamma}^{(y)} = \boldsymbol{T}_y \boldsymbol{z} \tag{3.52}$$

[6]Note that, since the role of y in $\widehat{l}^{(y)}(\boldsymbol{\gamma})$ is merely to ignore the yth component of the input $\boldsymbol{\gamma}$, we could have instead written $\widehat{l}^{(y)}(\boldsymbol{\gamma}) = \widehat{l}(\boldsymbol{J}_y \gamma)$ for y-invariant $\widehat{l}(\cdot)$ and $\boldsymbol{J}_y$ constructed by removing the yth row from the identity matrix.

with

$$\boldsymbol{T}_y = \begin{bmatrix} -\boldsymbol{I}_{y-1} & \boldsymbol{1}_{(y-1)\times 1} & \boldsymbol{0}_{(y-1)\times(K-y)} \\ \boldsymbol{0}_{1\times(y-1)} & 1 & \boldsymbol{0}_{1\times(K-y)} \\ \boldsymbol{0}_{(K-y)\times(y-1)} & \boldsymbol{1}_{(K-y)\times 1} & -\boldsymbol{I}_{K-y} \end{bmatrix} \tag{3.53}$$

to get (since $\det(\boldsymbol{T}_y) = 1$)

$$S_{ki}^{(y)} = \int_{\mathbb{R}^K} \Big(\boldsymbol{e}_k^{\mathsf{T}} \boldsymbol{T}_y^{-1} \boldsymbol{\gamma}\Big)^i \, l^{(y)}(\boldsymbol{\gamma}) \mathcal{N}\Big(\boldsymbol{\gamma}; \boldsymbol{T}_y \widehat{\boldsymbol{p}}, \boldsymbol{T}_y \boldsymbol{Q}^{\mathsf{p}} \boldsymbol{T}_y^{\mathsf{T}}\Big) \, \mathrm{d}\boldsymbol{\gamma}. \tag{3.54}$$

Then, applying the approximation (3.50) and

$$\begin{aligned} \mathcal{N}\Big(\boldsymbol{\gamma}; \boldsymbol{T}_y \widehat{\boldsymbol{p}}, \boldsymbol{T}_y \boldsymbol{Q}^{\mathsf{p}} \boldsymbol{T}_y^{\mathsf{T}}\Big) &= \mathcal{N}\Big(\gamma_y; \widehat{p}_y, q_y^{\mathsf{p}}\Big) \\ &\qquad \times \prod_{d\neq y} \mathcal{N}\Big(\gamma_d; \gamma_y - \widehat{p}_d, q_d^{\mathsf{p}}\Big) \end{aligned} \tag{3.55}$$

to (3.54), we find that

$$\begin{aligned} S_{ki}^{(y)} \approx \sum_{l=1}^{L} \alpha_l \int_{\mathbb{R}} \mathcal{N}\Big(\gamma_y; \widehat{p}_y, q_y^{\mathsf{p}}\Big) \bigg[\int_{\mathbb{R}^{K-1}} \Big(\boldsymbol{e}_k^{\mathsf{T}} \boldsymbol{T}_y^{-1} \boldsymbol{\gamma}\Big)^i \\ \times \prod_{d\neq y} \mathcal{N}\Big(\gamma_d; \gamma_y - \widehat{p}_d, q_d^{\mathsf{p}}\Big) \Phi\bigg(\frac{\gamma_d - \mu_{dl}}{\sigma_{dl}}\bigg) \, \mathrm{d}\gamma_d \bigg] \, \mathrm{d}\gamma_y. \end{aligned} \tag{3.56}$$

Noting that $\boldsymbol{T}_y^{-1} = \boldsymbol{T}_y$, we have

$$\boldsymbol{e}_k^{\mathsf{T}} \boldsymbol{T}_y^{-1} \boldsymbol{\gamma} = \begin{cases} \gamma_y - \gamma_k & k \neq y \\ \gamma_y & k = y \end{cases}. \tag{3.57}$$

Thus, for a fixed value of $\gamma_y = c$, the inner integral in (3.56) can be expressed as a product of linear combinations of terms

$$\int_{\mathbb{R}} \gamma^i \, \mathcal{N}\Big(\gamma; c - \widehat{p}, q\Big) \Phi\bigg(\frac{\gamma - \mu}{\sigma}\bigg) \, \mathrm{d}\gamma \triangleq T_i \tag{3.58}$$

with $i \in \{0, 1, 2\}$, which can be computed in closed form. In particular, defining $x \triangleq \frac{c - \widehat{p} - \mu}{\sqrt{\sigma^2 + q}}$, we have

$$T_0 = \Phi(x) \tag{3.59}$$

$$T_1 = (c - \widehat{p})\Phi(x) + \frac{q\phi(x)}{\sqrt{\sigma^2 + q}} \tag{3.60}$$

$$T_2 = \frac{(T_1)^2}{\Phi(x)} + q\Phi(x) - \frac{q^2 \phi(x)}{\sigma^2 + q}\Big(x + \frac{\phi(x)}{\Phi(x)}\Big), \tag{3.61}$$

which can be obtained using the results in [47, §3.9]. The outer integral in (3.56) can then be approximated via numerical integration.

If a grid of T values is used for numerical integration over γ_y in (3.56), then the overall complexity of the method grows as $O(MKLT)$. Our experiments indicate that relatively small values (e.g., $L = 2$ and $T = 7$) suffice.

Performance comparison

Above we described four methods of approximating lines 8-9 in Table 2.1 under diagonal $\boldsymbol{Q}^{\mathsf{p}}$ and $\boldsymbol{Q}^{\mathsf{z}}$. We now compare the accuracy and complexity of these methods. In particular, we measured the accuracy of the conditional mean (i.e., line 8) approximation as follows (for a given $\widehat{\boldsymbol{p}}$ and $\boldsymbol{Q}^{\mathsf{p}}$):

1. generate i.i.d. samples $\boldsymbol{z}_{\mathsf{true}}[t] \sim \mathcal{N}(\boldsymbol{z}; \widehat{\boldsymbol{p}}, \boldsymbol{Q}^{\mathsf{p}})$ and $y_{\mathsf{true}}[t] \sim p_{\mathsf{y}|\mathbf{z}}(y \,|\, \boldsymbol{z}_{\mathsf{true}}[t])$ for $t = 1 \dots T$,

2. compute the approximation $\widehat{\boldsymbol{z}}[t] \approx \mathrm{E}\{\mathbf{z} \,|\, \mathsf{y} = y_{\mathsf{true}}[t], \mathbf{p} = \widehat{\boldsymbol{p}}; \boldsymbol{Q}^{\mathsf{p}}\}$ using each method described in Sections 3.3.2–3.3.2,

3. compute average MSE $\triangleq \frac{1}{T}\sum_{t=1}^{T} \left\| \boldsymbol{z}_{\mathsf{true}}[t] - \widehat{\boldsymbol{z}}[t] \right\|_2^2$ for each method,

and we measured the combined runtime of lines 8-9 for each method. Unless otherwise noted, we used $K = 4$ classes, $\widehat{\boldsymbol{p}} = \boldsymbol{e}_1$, $\boldsymbol{Q}^{\mathsf{p}} = q^{\mathsf{p}}\boldsymbol{I}_K$, and $q^{\mathsf{p}} = 1$ in our experiments. For numerical integration (NI), we used a grid of size $T = 7$ and radius of $\alpha = 4$ standard deviations; for importance sampling (IS), we used $T = 1500$ samples; and for the Gaussian-mixture (GM) method, we used $L = 2$ mixture components and a grid size of $T = 7$. Empirically, we found that smaller grids or fewer samples compromised accuracy, whereas larger grids or more samples compromised runtime.

Figure 3.2 plots the normalized MSE versus variance q^{p} for the four methods under test, in addition to the trivial method $\widehat{\boldsymbol{z}}[t] = \widehat{\boldsymbol{p}}$. The figure shows that the NI, IS, and GM methods performed similarly across the full range of q^{p} and always outperform the trivial method. The Taylor-series method, however, breaks down when $q^{\mathsf{p}} > 1$. A close examination of the figure reveals that GM gave the best accuracy, IS the second best accuracy, and NI the third best accuracy.

Figure 3.3 shows the cumulative runtime (over $M = 500$ training samples) of the methods from Sections 3.3.2–3.3.2 versus the number of classes, K. Although the Taylor-series method was the fastest, we saw in Figure 3.2 that it is accurate only at small variances q^{p}. Figure 3.3 then shows GM was about an order-of-magnitude faster than IS, which was several orders-of-magnitude faster than NI.

Together, Figures 3.2-3.3, show that our proposed GM method dominated the IS and NI methods in both accuracy and runtime. Thus, for the remainder of the paper, we implement sum-product SHyGAMP using the GM method from Section 3.3.2.

3.3.3 Min-sum SHyGAMP: Inference of $\boldsymbol{x}_n$

With diagonal $\boldsymbol{Q}^{\mathsf{r}}_n$ and $\boldsymbol{Q}^{\mathsf{x}}_n$, the implementation of lines 16-17 in Table 2.1 can be significantly simplified. Recall that, when the prior p_{x} is chosen as i.i.d. Laplace (3.5), line 16 manifests as (3.21), which is in general a non-trivial optimization problem. But with diagonal $\boldsymbol{Q}^{\mathsf{r}}_n$, (3.21) decouples into K instances of the scalar optimization

$$x_{nk} = \arg\max_x -\frac{1}{2}\frac{(x-\widehat{r}_{nk})^2}{q^{\mathsf{r}}_{nk}} - \lambda|x|, \tag{3.62}$$

which is known to have the closed-form "soft thresholding" solution

$$\widehat{x}_{nk} = \operatorname{sgn}(\widehat{r}_{nk})\max\{0, |\widehat{r}_{nk}| - \lambda q^{\mathsf{r}}_{nk}\}. \tag{3.63}$$

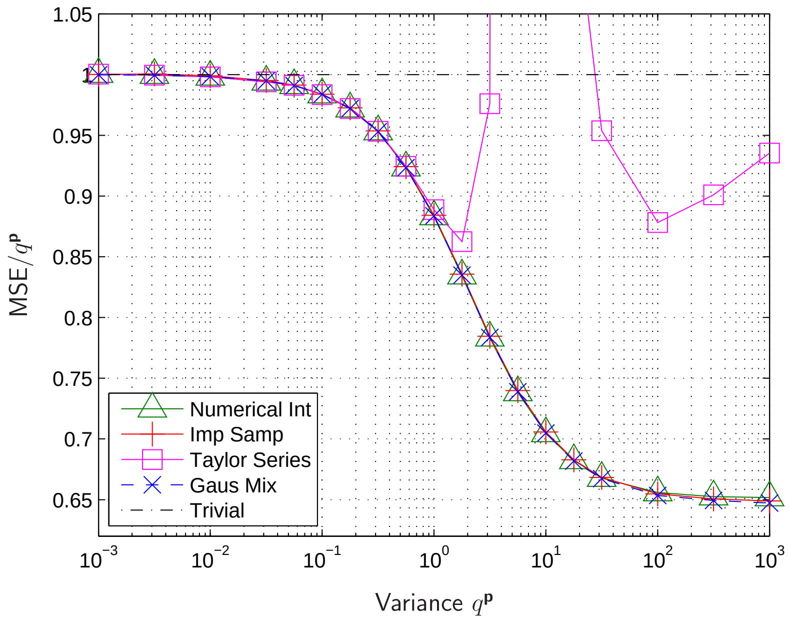


Figure 3.2: MSE/q^{p} versus variance q^{p} for various methods to compute line 8 in Table 2.1. Each point represents the average of 5×10^6 independent trials.

Above, $\operatorname{sgn}(r) = 1$ when $r \geq 0$ and $\operatorname{sgn}(r) = -1$ when $r < 0$.

Meanwhile, line 17 reduces to

$$q^{\mathsf{x}}_{nk} = \left[\frac{\partial^2}{\partial x^2} \left(\frac{1}{2} \frac{(x - \widehat{r}_{nk})^2}{q^{\mathsf{r}}_{nk}} + \lambda |x| \right) \bigg|_{x = \widehat{x}_{nk}} \right]^{-1}, \tag{3.64}$$

which equals q^{r}_{nk} when $\widehat{x}_{nk} \neq 0$ and is otherwise undefined. When $\widehat{x}_{nk} = 0$, we set $q^{\mathsf{x}}_{nk} = 0$.

3.3.4 Min-sum SHyGAMP: Inference of $\boldsymbol{z}_m$

With diagonal $\boldsymbol{Q}^{\mathsf{p}}_m$ and $\boldsymbol{Q}^{\mathsf{z}}_m$, the implementation of lines 5-6 in Table 2.1 also simplifies. Recall that, when the likelihood $p_{\mathsf{y}|\mathsf{z}}$ takes the MLR form in (3.3), line 5 manifests as (3.26), which can be solved using a component-wise Newton's method as in (3.31)-(3.33) for any $\boldsymbol{Q}^{\mathsf{p}}_m$ and $\boldsymbol{Q}^{\mathsf{z}}_m$. When $\boldsymbol{Q}^{\mathsf{p}}_m$ is diagonal, the first and second

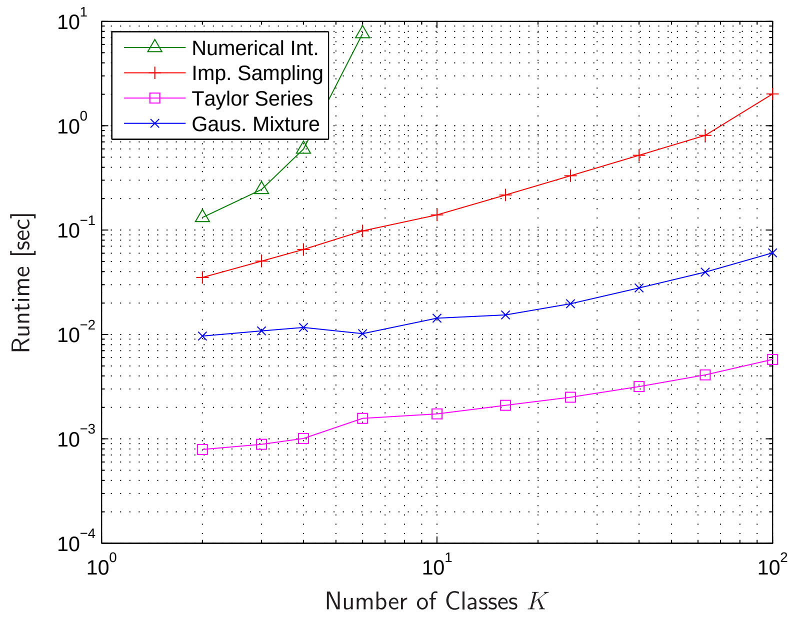


Figure 3.3: Cumulative runtime (over $M = 500$ samples) versus number-of-classes K for various methods to compute lines 8-9 in Table 2.1. Each point represents the average of 2000 independent trials.

derivatives (3.32)-(3.33) reduce to

$$g_{mk}^{(k)} = p_{\mathsf{y}|\mathbf{z}}(k|\widehat{\boldsymbol{z}}_m^{(t)}) - \delta_{y_m - k} + (\widehat{z}_{mk}^{(t)} - \widehat{p}_{mk})/q_{mk}^{\mathbf{p}}. \tag{3.65}$$

$$H_{mk}^{(t)} = p_{\mathsf{y}|\mathbf{z}}(k|\widehat{\boldsymbol{z}}_m^{(t)})^2 - p_{\mathsf{y}|\mathbf{z}}(k|\widehat{\boldsymbol{z}}_m^{(t)}) - 1/q_{mk}^{\mathbf{p}}, \tag{3.66}$$

which leads to a reduction in complexity.

Furthermore, line 6 simplifies, since with diagonal $\boldsymbol{Q}_m^{\mathbf{z}}$ it suffices to compute only the diagonal components of $\boldsymbol{H}_m^{(t)}$ in (3.30). In particular, when $\boldsymbol{Q}_m^{\mathbf{p}}$ is diagonal, the result becomes

$$q_{mk}^{\mathbf{z}} = \frac{1}{1/q_{mk}^{\mathbf{p}} + p_{\mathsf{y}|\mathbf{z}}(k|\widehat{\boldsymbol{z}}_m) - p_{\mathsf{y}|\mathbf{z}}(k|\widehat{\boldsymbol{z}}_m)^2}. \tag{3.67}$$

Algorithm	Quantity	Method	Complexity
SPA-SHyGAMP	$\widehat{\boldsymbol{x}}$	CF	$O(K)$
	$\boldsymbol{Q}^{\mathsf{x}}$	CF	$O(K)$
	$\widehat{\boldsymbol{z}}$	GM	$O(KLT)$
	$\boldsymbol{Q}^{\mathsf{z}}$	GM	$O(KLT)$
MSA-SHyGAMP	$\widehat{\boldsymbol{x}}$	ST	$O(D)$
	$\boldsymbol{Q}^{\mathsf{x}}$	CF	$O(K)$
	$\widehat{\boldsymbol{z}}$	CWN	$O(KT)$
	$\boldsymbol{Q}^{\mathsf{z}}$	CF	$O(K^3)$

Table 3.2: A summary of the K-dimensional inference sub-problems encountered when running SPA-SHyGAMP or MSA-SHyGAMP, as well as their associated computational costs. 'CF' = 'closed form', 'GM' = 'Gaussian mixture', 'ST' = 'Soft-thresholding', and 'CWN' = 'component-wise Newton's method'. For the GM, L denotes the number of mixture components and T the number of samples in the 1D numerical integral, and for CWN T denotes the number of iterations.

Algorithm	HyGAMP	SHyGAMP
Diagonal covariance matrices		✓
Simplified K-dimensional inference		✓
Scalar-variance approximation		✓
Online parameter tuning		✓

Table 3.3: High-level comparison of SHyGAMP and HyGAMP.

3.3.5 SHyGAMP summary

In summary, by approximating the covariance matrices as diagonal, the SPA-SHyGAMP and MSA-SHyGAMP algorithms improve computationally upon their HyGAMP counterparts. A summary of the K-dimensional inference problems encountered when running SPA-SHyGAMP or MSA-SHyGAMP, as well as their associated computational costs, is given in Table 3.2. A high-level comparison between HyGAMP and SHyGAMP is given in Table 3.3.

3.4 Online Parameter Tuning

The weight vector priors in (3.5) and (3.35) depend on modeling parameters that, in practice, must be tuned. Although cross-validation (CV) is the customary approach to tuning such model parameters, it can be very computationally costly, since each parameter must be tested over a grid of hypothesized values and over multiple data folds. For example, K-fold[7] cross-validation tuning of P parameters using G hypothesized values of each parameter requires the training and evaluation of KG^P classifiers.

3.4.1 Parameter selection for Sum-product SHyGAMP

For SPA-SHyGAMP, we propose to use the zero-mean Bernoulli-Gaussian prior in (3.35), which has parameters β_k, m_k, and v_k. Instead of CV, we use the EM-GM-AMP framework described in [4] to tune these parameters online. See [14] for details regarding the initialization of β_k, m_k, and v_k.

3.4.2 Parameter selection for Min-sum SHyGAMP

To use MSA-SHyGAMP with the Laplacian prior in (3.5), we need to specify the scale parameter λ. For this, we use a modification of the SURE-AMP framework from [35], which adjusts λ to minimize the Stein's unbiased risk estimate (SURE) of the weight-vector MSE.

We describe our method by first reviewing SURE and SURE-AMP. First, suppose that the goal is to estimate the value of x, which is a realization of the random variable

[7]This K has no relation to the number of classes.

x, from the noisy observation r, which is a realization of

$$\mathsf{r} = \mathsf{x} + \sqrt{q^{\mathsf{r}}}\mathsf{w}, \tag{3.68}$$

with $\mathsf{w} \sim \mathcal{N}(0,1)$ and $q^{\mathsf{r}} > 0$. For this purpose, consider an estimate of the form $\widehat{x} = f(r, q^{\mathsf{r}}; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ contains tunable parameters. For convenience, define the shifted estimation function $g(r, q^{\mathsf{r}}; \boldsymbol{\theta}) \triangleq f(r, q^{\mathsf{r}}; \boldsymbol{\theta}) - r$ and its derivative $g'(r, q^{\mathsf{r}}; \boldsymbol{\theta}) \triangleq \frac{\partial}{\partial r} g(r, q^{\mathsf{r}}; \boldsymbol{\theta})$. Then Stein [48] established the following result on the mean-squared error, or risk, of the estimate $\widehat{x}$:

$$\mathrm{E}\left\{[\widehat{\mathsf{x}} - \mathsf{x}]^2\right\} = q^{\mathsf{r}} + \mathrm{E}\left\{g^2(\mathsf{r}, q^{\mathsf{r}}; \boldsymbol{\theta}) + 2q^{\mathsf{r}} g'(\mathsf{r}, q^{\mathsf{r}}; \boldsymbol{\theta})\right\}. \tag{3.69}$$

The implication of (3.69) is that, given only the noisy observation r and the noise variance q^{r}, one can compute an estimate

$$\mathrm{SURE}(r, q^{\mathsf{r}}; \boldsymbol{\theta}) \triangleq q^{\mathsf{r}} + g^2\Big(r, q^{\mathsf{r}}; \boldsymbol{\theta}) + 2q^{\mathsf{r}} g'(r, q^{\mathsf{r}}; \boldsymbol{\theta}) \tag{3.70}$$

of the $\mathrm{MSE}(\boldsymbol{\theta}) \triangleq \mathrm{E}\left\{[\widehat{\mathsf{x}} - \mathsf{x}]^2\right\}$ that is unbiased, i.e.,

$$\mathrm{E}\left\{\mathrm{SURE}(\mathsf{r}, q^{\mathsf{r}}; \boldsymbol{\theta})\right\} = \mathrm{MSE}(\boldsymbol{\theta}). \tag{3.71}$$

These unbiased risk estimates can then be used as a surrogate for the true MSE when tuning $\boldsymbol{\theta}$.

In [35], it was noticed that the assumption (3.68) is satisfied by AMP's denoiser inputs $\{\widehat{r}_n\}_{n=1}^N$, and thus [35] proposed to tune the soft threshold λ to minimize the SURE:

$$\widehat{\lambda} = \arg\min_{\lambda} \sum_{n=1}^N g^2\Big(\widehat{r}_n, q^{\mathsf{r}}; \lambda) + 2q^{\mathsf{r}} g'(\widehat{r}_n, q^{\mathsf{r}}; \lambda). \tag{3.72}$$

Recalling the form of the estimator $f(\cdot)$ from (3.63), we have

$$g^2(\widehat{r}_n, q^{\mathsf{r}}; \lambda) = \begin{cases} \lambda^2 (q^{\mathsf{r}})^2 & \text{if } |\widehat{r}_n| > \lambda q^{\mathsf{r}} \\ \widehat{r}_n^2 & \text{otherwise} \end{cases} \tag{3.73}$$

$$g'(\widehat{r}_n, q^{\mathsf{r}}; \lambda) = \begin{cases} -1 & \text{if } |\widehat{r}_n| < \lambda q^{\mathsf{r}} \\ 0 & \text{otherwise} \end{cases}. \tag{3.74}$$

However, solving (3.72) for λ is non-trivial because the objective is non-smooth and has many local minima. A stochastic gradient descent approach was proposed in [35], but its convergence speed is too slow to be practical.

Since (3.68) also matches the scalar-variance SHyGAMP model from Section 2.4, we propose to use SURE to tune λ for min-sum SHyGAMP. But, instead of the empirical average in (3.72), we propose to use a statistical average, i.e.,

$$\widehat{\lambda} = \arg\min_{\lambda} \underbrace{\mathrm{E}\Big\{ g^2\Big(\mathsf{r}, q^{\mathsf{r}}; \lambda) + 2 q^{\mathsf{r}} g'(\mathsf{r}, q^{\mathsf{r}}; \lambda)\Big\}}_{\triangleq\, J(\lambda)}, \tag{3.75}$$

by modeling the random variable r as a Gaussian mixture (GM) whose parameters are fitted to $\{\widehat{r}_{nk}\}$. As a result, the objective in (3.75) is smooth. Moreover, by constraining the smallest mixture variance to be at least q^{r}, the objective becomes unimodal, in which case $\widehat{\lambda}$ from (3.75) is the unique root of $\frac{\mathrm{d}}{\mathrm{d}\lambda} J(\lambda)$. To find this root, we use the bisection method. In particular, due to (3.73)-(3.74), the objective in (3.75) becomes

$$\begin{aligned} J(\lambda) = &\int_{-\infty}^{-\lambda q^{\mathsf{r}}} p_{\mathsf{r}}(r) \lambda^2 (q^{\mathsf{r}})^2 \,\mathrm{d}r + \int_{-\lambda q^{\mathsf{r}}}^{\lambda q^{\mathsf{r}}} p_{\mathsf{r}}(r)(r^2 - 2q^{\mathsf{r}}) \,\mathrm{d}r \\ &+ \int_{\lambda q^{\mathsf{r}}}^{\infty} p_{\mathsf{r}}(r) \lambda^2 (q^{\mathsf{r}})^2 \,\mathrm{d}r, \end{aligned} \tag{3.76}$$

from which it can be shown that [14]

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\lambda} J(\lambda) = &\, 2\lambda (q^{\mathsf{r}})^2 \Big[1 - \Pr\{-\lambda q^{\mathsf{r}} < \mathsf{r} < \lambda q^{\mathsf{r}}\}\Big] \\ &- \Big[p_{\mathsf{r}}(\lambda q^{\mathsf{r}}) + p_{\mathsf{r}}(-\lambda q^{\mathsf{r}})\Big] 2 (q^{\mathsf{r}})^2. \end{aligned} \tag{3.77}$$

For GM fitting, we use the standard EM approach [23] and find that relatively few (e.g., $L = 3$) mixture terms suffice. Note that we re-tune λ using the above technique at each iteration of Table 2.1, immediately before line 16. Experimental verification of our method is provided in Section 3.5.2.

3.5 Numerical Experiments

In this section we describe the results of several experiments used to test SHyGAMP. In these experiments, EM-tuned SPA-SHyGAMP and SURE-tuned MSA-SHyGAMP were compared to two state-of-the-art sparse MLR algorithms: SBMLR [27] and GLMNET [26]. We are particularly interested in SBMLR and GLMNET because [26,27] show that they have strong advantages over earlier algorithms, e.g., [10,24,25]. As described in Section 3.1.2, both SBMLR and GLMNET use ℓ_1 regularization, but SBMLR tunes the regularization parameter λ using evidence maximization while GLMNET tunes it using cross-validation (using the default value of 10 folds unless otherwise noted). For SBMLR and GLMNET, we ran code written by the authors [8] [9] under default settings (unless otherwise noted). For SHyGAMP, we used the damping modification described in [39]. We note that the runtimes reported for all algorithms include the total time spent to tune all parameters and train the final classifier.

Due to space limitations, we do not show the performance of the more complicated HyGAMP algorithm from Section 3.2. However, our experience suggests that HyGAMP generates weight matrices $\widehat{\boldsymbol{X}}$ that are very similar to those generated by SHyGAMP, but with much longer runtimes, especially as K grows.

[8]SBMLR obtained from http://theoval.cmp.uea.ac.uk/matlab/

[9]GLMNET obtained from http://www.stanford.edu/~hastie/glmnet_matlab/

3.5.1 Synthetic data in the $M \ll N$ regime

We first describe the results of three experiments with synthetic data. For these experiments, the training data were randomly generated and algorithm performance was averaged over several data realizations. In all cases, we started with balanced training labels $y_m \in \{1, \ldots, K\}$ for $m = 1, \ldots, M$ (i.e., M/K examples from each of K classes). Then, for each data realization, we generated M i.i.d. training features $\boldsymbol{a}_m$ from the class-conditional generative distribution $\boldsymbol{a}_m \,|\, y_m \sim \mathcal{N}(\boldsymbol{\mu}_{y_m}, v\boldsymbol{I}_N)$. In doing so, we chose the intra-class variance, v, to attain a desired Bayes error rate (BER) of 10% (see [14] for details), and we used randomly generated S-sparse orthonormal class means, $\boldsymbol{\mu}_k \in \mathbb{R}^N$. In particular, we generated $[\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K]$ by drawing a $S \times S$ matrix with i.i.d. $\mathcal{N}(0, 1)$ entries, performing a singular value decomposition, and zero-padding the first K left singular vectors to length N. We note that our generation of $\boldsymbol{y}, \boldsymbol{A}, \boldsymbol{X}$ is matched [49] to the multinomial logistic model (3.2)-(3.3).

Given a training data realization, each algorithm was invoked to yield a weight matrix $\widehat{\boldsymbol{X}} = [\widehat{\boldsymbol{x}}_1, \ldots, \widehat{\boldsymbol{x}}_K]$. The corresponding expected test-error rate was then analytically computed as

$$\Pr\{\text{err}\} = 1 - \frac{1}{K} \sum_{y=1}^{K} \Pr\{\text{cor}|y\} \tag{3.78}$$

$$\Pr\{\text{cor}|y\} = \Pr \bigcap_{k \neq y} \left\{ (\widehat{\boldsymbol{x}}_y - \widehat{\boldsymbol{x}}_k)^{\mathsf{T}} \mathbf{a} < (\widehat{\boldsymbol{x}}_y - \widehat{\boldsymbol{x}}_k)^{\mathsf{T}} \boldsymbol{\mu}_y \right\}, \tag{3.79}$$

where $\mathbf{a} \sim \mathcal{N}(\mathbf{0}, v\boldsymbol{I}_N)$ and the multivariate normal cdf in (3.79) was computed using Matlab's `mvncdf`.

For all three synthetic-data experiments, we used $K = 4$ classes and $S \ll M \ll N$. In the first experiment, we fixed S and N and we varied M; in the second experiment, we fixed N and M and we varied S; and in the third experiment, we fixed S and M

Experiment	M	N	S	K
1	$\{100, \ldots, 5000\}$	10000	10	4
2	300	30000	$\{5, \ldots, 30\}$	4
3	200	$\{10^3, \ldots, 10^{5.5}\}$	10	4
4	300	30000	25	4

Table 3.4: Configurations of the synthetic-data experiments.

and we varied N. The specific values/ranges of S, M, N used for each experiment are given in Table 3.4.

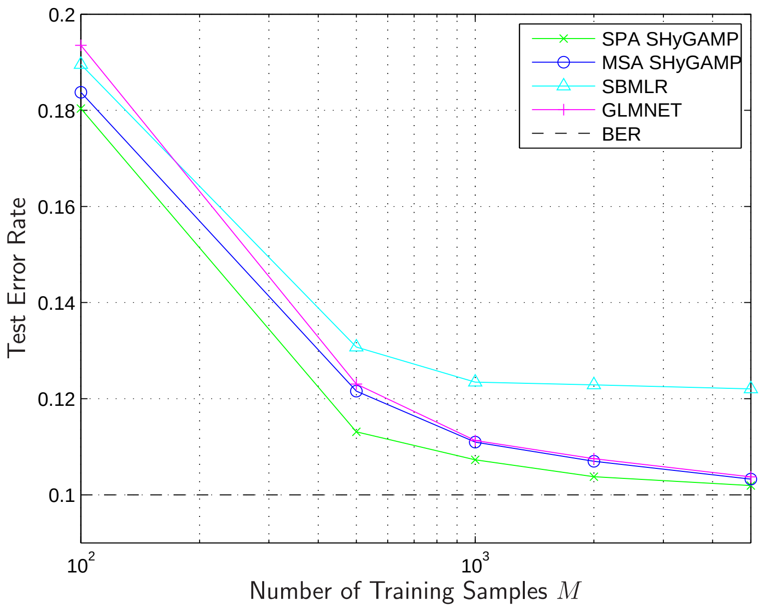
Figure 3.4 shows the expected test-error rate and runtime, respectively, versus the number of training examples, M, averaged over 12 independent trials. Figure 3.4a shows that, at all tested values of M, SPA-SHyGAMP gave the best error-rates and MSA-SHyGAMP gave the second best error-rates, although those reached by GLMNET were similar at large M. Moreover, the error-rates of SPA-SHyGAMP, MSA-SHyGAMP, and GLMNET all converged towards the BER as M increased, whereas that of SBMLR did not. Since MSA-SHyGAMP, GLMNET, and SBMLR all solve the same ℓ_1-regularized MLR problem, the difference in their error-rates can be attributed to the difference in their tuning of the regularization parameter λ. Figure 3.4b shows that, for $M > 500$, SPA-SHyGAMP was the fastest, followed by MSA-SHyGAMP, SBMLR, and GLMNET. Note that the runtimes of SPA-SHyGAMP, MSA-SHyGAMP, and GLMNET increased linearly with M, whereas the runtime of SBMLR increased quadratically with M.

Figure 3.5 shows the expected test-error rate and runtime, respectively, versus feature-vector sparsity, S, averaged over 12 independent trials. Figure 3.5a shows that, at all tested values of S, SPA-SHyGAMP gave the best error-rates and MSA-SHyGAMP gave the second best error-rates. Figure 3.5b shows that SPA-SHyGAMP and MSA-SHyGAMP gave the fastest runtimes. All runtimes were approximately invariant to S.

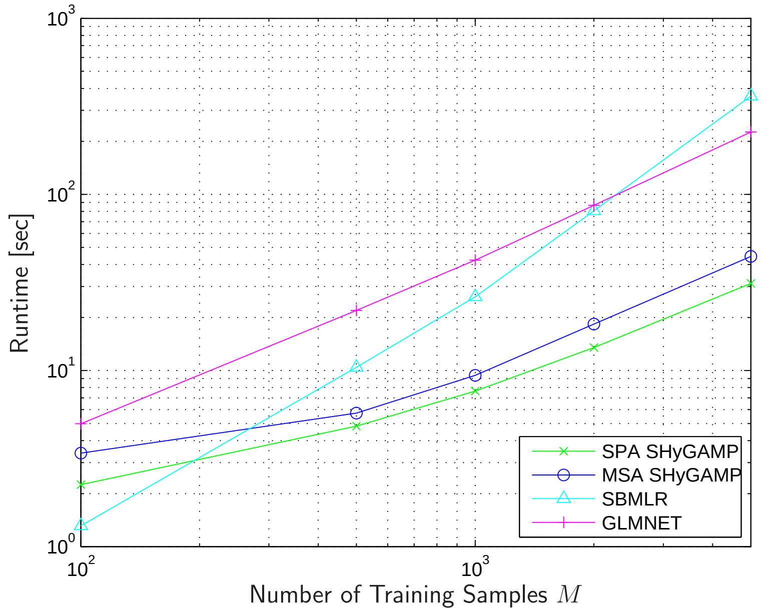
Figure 3.6 shows the expected test-error rate and runtime, respectively, versus the number of features, N, averaged over 12 independent trials. Figure 3.6a shows that, at all tested values of N, MSA-SHyGAMP gave lower error-rates than SBMLR and GLMNET. Meanwhile, SPA-SHyGAMP gave the lowest error-rates for certain values of N. Figure 3.6b shows that SPA-SHyGAMP and MSA-SHyGAMP gave the fastest runtimes for $N \geq 10000$, while SBMLR gave the fastest runtimes for $N \leq 3000$. All runtimes increased linearly with N.

3.5.2 Example of SURE tuning

Although the good error-rate performance of MSA-SHyGAMP in Section 3.5.1 suggests that the SURE λ-tuning method from Section 3.4.2 is working reliably, we now describe a more direct test of its behavior. Using synthetic data generated as described in Section 3.5.1 with $K = 4$ classes, $N = 30000$ features, $M = 300$ examples, and sparsity $S = 25$, we ran MSA-SHyGAMP using various fixed values of λ. In the sequel, we refer to this experiment as "Synthetic Experiment 4." The resulting expected test-error rate versus λ (averaged over 10 independent realizations) is shown in Figure 3.7. For the same realizations, we ran MSE-SHyGAMP with SURE-tuning and plot the resulting error-rate and average $\widehat{\lambda}$ in Figure 3.7. From Figure 3.7, we see that the SURE λ-tuning method matched both the minimizer and the minimum of the error-versus-λ trace of fixed-λ MSA-SHyGAMP.

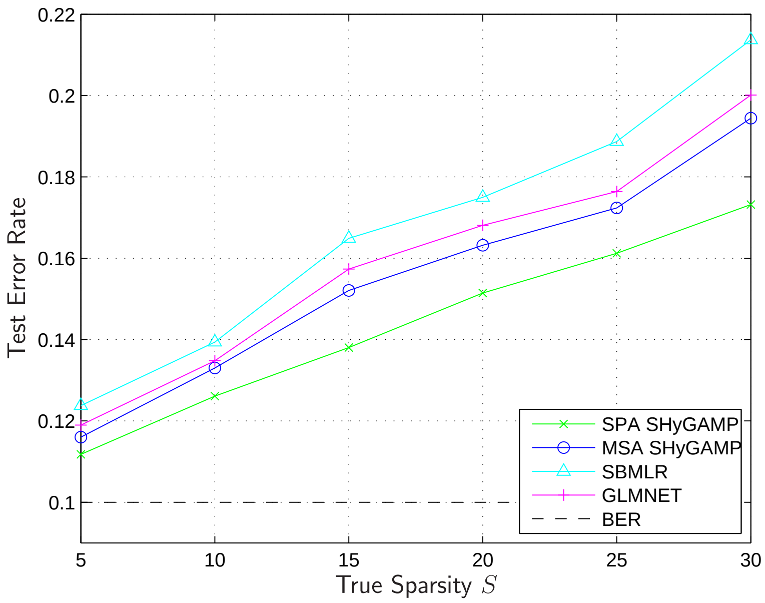


(a) Error

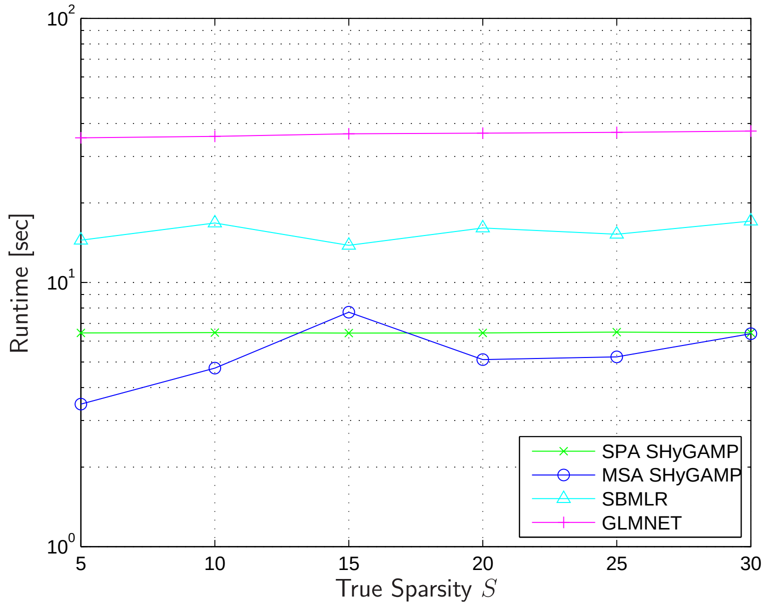


(b) Runtime

Figure 3.4: Synthetic Experiment 1: expected test-error rate and runtime versus M. Here, $K = 4$, $N = 10000$, and $S = 10$.

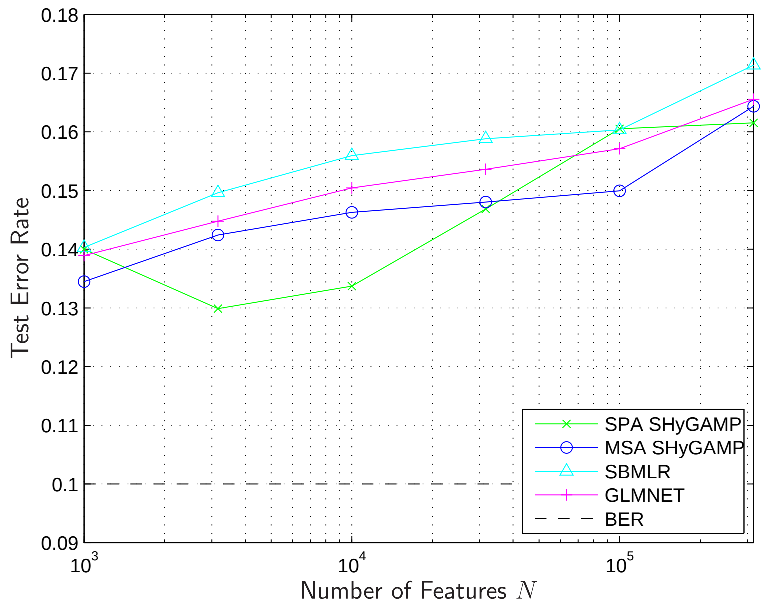


(a) Error

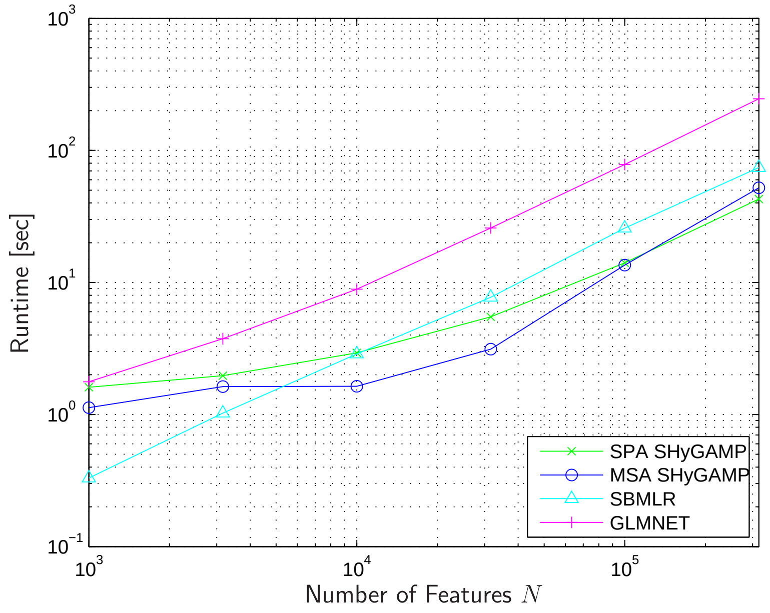


(b) Runtime

Figure 3.5: Synthetic Experiment 2: expected test-error rate and runtime versus S. Here, $K = 4$, $M = 300$, and $N = 30000$.



(a) Error



(b) Runtime

Figure 3.6: Synthetic Experiment 3: expected test-error rate and runtime versus N. Here, $K = 4$, $M = 200$, and $S = 10$.

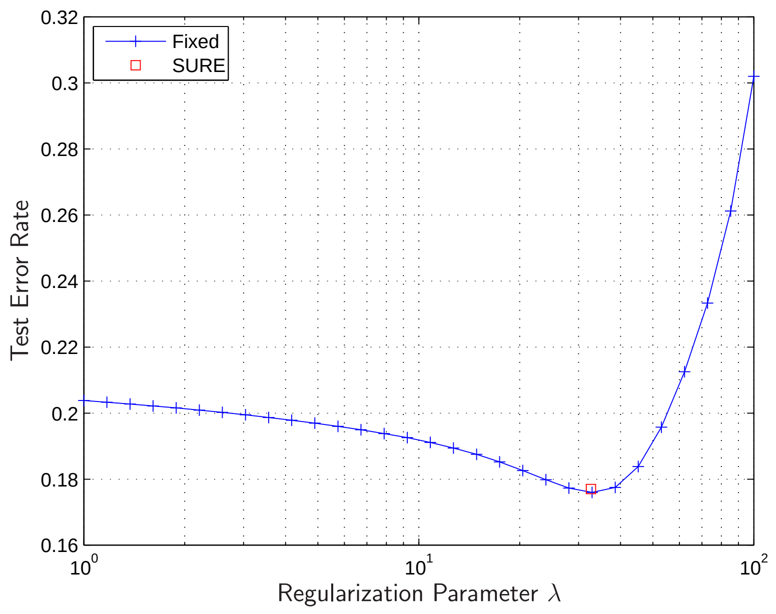


Figure 3.7: Synthetic experiment 4: expected test-error rate versus regularization parameter λ for fixed-λ MSA-SHyGAMP. Here, $K = 4$, $M = 300$, $N = 30000$, and $S = 25$. Also shown is the average test-error rate for SURE-tuned MSA-SHyGAMP plotted at the average value of $\widehat{\lambda}$.

3.5.3 Micro-array gene expression

Next we consider classification and feature-selection using micro-array gene expression data. Here, the labels indicate which type of disease is present (or no disease) and the features represent gene expression levels. The objective is i) to determine which subset of genes best predicts the various diseases and ii) to classify whether an (undiagnosed) patient is at risk for any of these diseases based on their gene profile.

We tried two datasets: one from Sun et al. [15] and one from Bhattacharjee et al. [16]. The Sun dataset includes $M = 179$ examples, $N = 54613$ features, and $K = 4$ classes; and the Bhattacharjee dataset includes $M = 203$ examples, $N = 12600$

features, and $K = 5$ classes. With the Sun dataset, we applied a $\log_2(\cdot)$ transformation and z-scored prior to processing, while with Bhattacharjee we simply z-scored (since the dataset included negative values).

The test-error rate was estimated as follows for each dataset. We consider a total of T "trials." For the tth trial, we i) partition the dataset into a training subset of size $M_{\mathsf{train},t}$ and a test subset of size $M_{\mathsf{test},t}$, ii) design the classifier using the training subset, and iii) apply the classifier to the test subset, recording the test errors $\{e_{tm}\}_{m=1}^{M_{\mathsf{test},t}}$, where $e_{tm} \in \{0,1\}$ indicates whether the mth example was in error. We then estimate the average test-error rate using the empirical average $\widehat{\mu} \triangleq M_{\mathsf{test}}^{-1} \sum_{t=1}^{T} \sum_{m=1}^{M_{\mathsf{test},t}} e_{tm}$, where $M_{\mathsf{test}} = \sum_{t=1}^{T} M_{\mathsf{test},t}$. If the test sets are constructed without overlap, we can model $\{e_{tm}\}$ as i.i.d. Bernoulli(μ), where μ denotes the true test-error rate. Then, since $\widehat{\mu}$ is Binomial(μ, M_{test}), the standard deviation (SD) of our error-rate estimate $\widehat{\mu}$ is $\sqrt{\text{var}\{\widehat{\mu}\}} = \sqrt{\mu(1-\mu)/M_{\mathsf{test}}}$. Since μ is unknown, we approximate the SD by $\sqrt{\widehat{\mu}(1-\widehat{\mu})/M_{\mathsf{test}}}$.

Tables 3.5 and 3.6 show, for each algorithm, the test-error rate estimate $\widehat{\mu}$, the approximate SD $\sqrt{\widehat{\mu}(1-\widehat{\mu})/M_{\mathsf{test}}}$ of the estimate, the average runtime, and two metrics for the sparsity of $\widehat{\boldsymbol{X}}$. The $\|\widehat{\boldsymbol{X}}\|_0$ metric quantifies the number of non-zero entries in $\widehat{\boldsymbol{X}}$ (i.e., absolute sparsity), while the $\widehat{S}_{99}$ metric quantifies the number of entries of $\widehat{\boldsymbol{X}}$ needed to reach 99% of the Frobenius norm of $\widehat{\boldsymbol{X}}$ (i.e., effective sparsity). We note that the reported values of $\widehat{S}_{99}$ and $\|\widehat{\boldsymbol{X}}\|_0$ represent the average over the T folds. For both the Sun and Bhattacharjee datasets, we used $T = 19$ trials and $M_{\mathsf{test},t} = \lfloor M/20 \rfloor\ \forall t$.

Table 3.5 shows results for the Sun dataset. There we see that MSA-SHyGAMP gave the best test-error rate, although the other algorithms were not far behind and all error-rate estimates were within the estimator standard deviation. SPA-SHyGAMP

Algorithm	% Error (SD)	Runtime (s)	$\widehat{S}_{99}$	$\|\widehat{\boldsymbol{X}}\|_0$
SPA-SHyGAMP	33.3 (3.8)	6.86	20.05	218 452
MSA-SHyGAMP	31.0 (3.7)	13.59	93.00	145.32
SBMLR	31.6 (3.7)	22.48	49.89	72.89
GLMNET	33.9 (3.8)	31.93	10.89	16.84

Table 3.5: Estimated test-error rate, standard deviation of estimate, runtime, and sparsities for the Sun dataset.

was the fastest algorithm and MSA-SHyGAMP was the second fastest, with the remaining algorithms running 3× to 5× slower than SPA-SHyGAMP. GLMNET's weights were the sparsest according to both sparsity metrics. SPA-SHyGAMP's weights had the second lowest value of $\widehat{S}_{99}$, even though they were technically non-sparse (i.e., $\|\widehat{\boldsymbol{X}}\|_0 = 218\,452 = NK$) as expected. Meanwhile, MSA-SHyGAMP's weights were the least sparse according to the $\widehat{S}_{99}$ metric.

Table 3.6 shows results for the Bhattacharjee dataset. In this experiment, SPA-SHyGAMP and SBMLR were tied for the best error rate, MSA-SHyGAMP was 0.5 standard-deviations worse, and GLMNET was 1.2 standard-deviations worse. However, SPA-SHyGAMP ran about twice as fast as SBMLR, and 4× as fast as GLMNET. As in the Sun dataset, SPA-SHyGAMP returned the sparsest weight matrix according to the $\widehat{S}_{99}$ metric. The sparsities of the weight matrices returned by the other three algorithms were similar to one another in both metrics. Unlike in the Sun dataset, MSA-SHyGAMP and SBMLR had similar runtimes (which is consistent with Figure 3.6b since N is lower here than in the Sun dataset).

Algorithm	% Error (SD)	Runtime (s)	$\widehat{S}_{99}$	$\|\widehat{\boldsymbol{X}}\|_0$
SPA-SHyGAMP	9.5 (2.1)	3.26	16.15	63 000
MSA-SHyGAMP	10.5 (2.2)	6.11	55.20	84.65
SBMLR	9.5 (2.1)	6.65	44.25	79.10
GLMNET	12.0 (2.4)	13.67	49.65	89.40

Table 3.6: Estimated test-error rate, standard deviation of estimate, runtime, and sparsities for the Bhattacharjee dataset.

3.5.4 Text classification with the RCV1 dataset

Next we consider text classification using the Reuter's Corpus Volume 1 (RCV1) dataset [20]. Here, each sample $(y_m, \boldsymbol{a}_m)$ represents a news article, where y_m indicates the article's topic and $\boldsymbol{a}_m$ indicates the frequencies of common words in the article. The version of the dataset that we used[10] contained $N = 47\,236$ features and 53 topics. However, we used only the first $K = 25$ of these topics (to reduce the computational demand). Also, we retained the default training and test partitions, which resulted in the use of $M = 14\,147$ samples for training and 469 571 samples for testing.

The RCV1 features are very sparse (only 0.326% of the features are non-zero) and non-negative, which conflicts with the standard assumptions used for the derivation of AMP algorithms: that $\boldsymbol{A}$ is i.i.d. zero-mean and sub-Gaussian. Interestingly, the RCV1 dataset also caused difficulties for SBMLR, which diverged under default settings. This divergence was remedied by decreasing the value of a step-size parameter[11] to 0.1 from the default value of 1.

[10]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html

[11]See the variable `scale` on lines 129 and 143 of `sbmlr.m`.

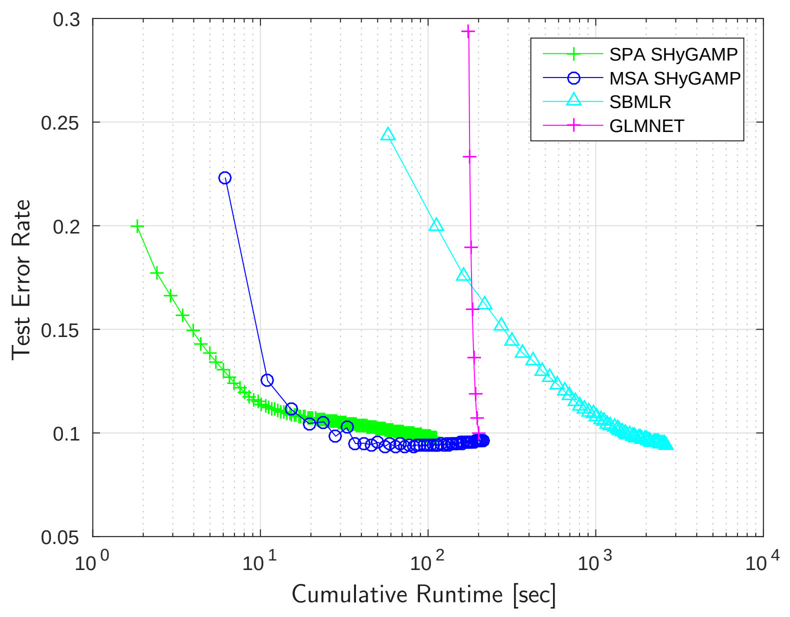


Figure 3.8: Test-error rate versus runtime for the RCV1 dataset.

Figure 3.8 shows test-error rate versus runtime for SPA-SHyGAMP, MSA-SHyGAMP, SBMLR, and GLMNET on the RCV1 dataset. In the case of SPA-SHyGAMP, MSA-SHyGAMP and SBMLR, each point in the figure represents one iteration of the corresponding algorithm. For GLMNET, each data-point represents one iteration of the algorithm *after* its cross-validation stage has completed.[12] We used 2 CV folds (rather than the default 10) in this experiment to avoid excessively long runtimes. The figure shows that the SHyGAMP algorithms converged more than an order-of-magnitude faster than SBMLR and GLMNET, although the final error rates were similar. SPA-SHyGAMP displayed faster initial convergence, but MSA-SHyGAMP eventually caught up.

[12] GLMNET spent most of its time on cross-validation. After cross-validation, GLMNET took 25.26 seconds to run, which is similar to the total runtimes of SPA-SHyGAMP and MSE-SHyGAMP.

3.5.5 MNIST handwritten digit recognition

Finally, we consider handwritten digit recognition using the Mixed National Institute of Standards and Technology (MNIST) dataset [50]. This dataset consists of 70 000 examples, where each example is an $N = 784$ pixel image of one of $K = 10$ digits between 0 and 9. These features were again non-negative, which conflicts with the standard AMP assumption of i.i.d. zero-mean $\boldsymbol{A}$.

Our experiment characterized test-error rate versus the number of training examples, M, for the SPA-SHyGAMP, MSA-SHyGAMP, SBMLR, and GLMNET algorithms. For each value of M, we performed 50 Monte-Carlo trials. In each trial, M training samples were selected uniformly at random and the remainder of the data were used for testing. Figure 3.9 shows the average estimated test-error rate $\widehat{\mu}$ versus the number of training samples, M, for the algorithms under test. The error-bars in the figure correspond to the average of the per-trial estimated SD over the 50 trials. For SBMLR, we reduced the stepsize to 0.5 from the default value of 1 to prevent a significant degradation of test-error rate. The figure shows SPA-SHyGAMP attaining significantly better error-rates than the other algorithms at small values of M (and again at the largest value of M considered for the plot). For this plot, M was chosen to focus on the $M < N$ regime.

3.6 Conclusion

For the problem of multi-class linear classification and feature selection, we proposed several AMP-based approaches to sparse multinomial logistic regression. We started by proposing two algorithms based on HyGAMP [13], one of which finds the

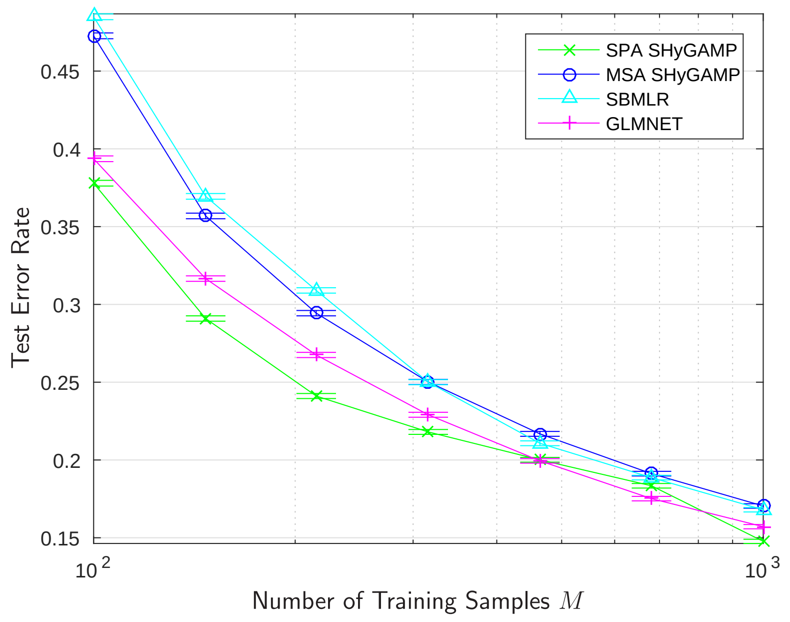


Figure 3.9: Estimated test-error rate versus M for the MNIST dataset, with error-bars indicating the standard deviation of the estimate.

maximum a posteriori (MAP) linear classifier based on the multinomial logistic likelihood and a Laplacian prior, and the other of which finds an approximation of the test-error-rate minimizing linear classifier based on the multinomial logistic likelihood and a Bernoulli-Gaussian prior. The numerical implementation of these algorithms is challenged, however, by the need to solve K-dimensional inference problems of multiplicity M at each HyGAMP iteration. Thus, we proposed simplified HyGAMP (SHyGAMP) approximations based on a diagonalization of the message covariances and a careful treatment of the K-dimensional inference problems. In addition, we described EM- and SURE-based methods to tune the hyperparameters of the assumed statistical model. Finally, using both synthetic and real-world datasets, we demonstrated improved error-rate and runtime performance relative to the state-of-the-art SBMLR [26] and GLMNET [27] approaches to sparse MLR.

Chapter 4: Sketched Clustering via Approximate Message Passing

4.1 Introduction

In this chapter[13] we consider the problem of sketched clustering. Given a dataset $\boldsymbol{D} \triangleq [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_T] \in \mathbb{R}^{N \times T}$ comprising T samples of dimension N, the standard clustering problem is to find K centroids $\boldsymbol{X} \triangleq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K] \in \mathbb{R}^{N \times K}$ that minimize the sum of squared errors (SSE)

$$\mathrm{SSE}(\boldsymbol{D}, \boldsymbol{X}) \triangleq \frac{1}{T} \sum_{t=1}^{T} \min_k \|\boldsymbol{d}_t - \boldsymbol{x}_k\|_2^2. \tag{4.1}$$

Finding the optimal $\boldsymbol{X}$ is an NP-hard problem [51]. Thus, many heuristic approaches have been proposed, such as the *k-means* algorithm [52,53]. Because k-means can get trapped in bad local minima, robust variants have been proposed, such as *k-means++* [54], which uses a careful random initialization procedure to yield solutions with SSE that have on average $\leq 8(\ln K + 2)$ times the minimal SSE. The computational complexity of k-means++ scales as $O(TKNI)$, with I the number of iterations, which is impractical when T is large.

[13]The content from this chapter is largely excerpted from the manuscript co-authored with Antoine Chatalic, Rémi Gribonval, and Philip Schniter, titled "Sketched Clustering via Hybrid Approximate Message Passing."

4.1.1 Sketched Clustering

In *sketched clustering* [55–57], the dataset $\boldsymbol{D}$ is first sketched down to a vector $\boldsymbol{y}$ with $M = O(KN)$ components, from which the centroids $\boldsymbol{X}$ are subsequently extracted. In the typical case that $K \ll T$, the sketch consumes much less memory than the original dataset. If the sketch can be performed efficiently, then—since the complexity of centroid-extraction is invariant to T—sketched clustering may be more efficient than direct clustering methods when T is large. Note, for example, that k-means++ processes the T data samples in $\boldsymbol{D}$ at every iteration, whereas sketched clustering processes the T data samples in $\boldsymbol{D}$ only once, during the sketching step.

In this work, we focus on sketches of the type proposed by Keriven et al. in [55,56], which use $\boldsymbol{y} = [y_1, \dots, y_M]^\mathsf{T}$ with

$$y_m = \frac{1}{T}\sum_{t=1}^{T} \exp(\mathrm{j}\boldsymbol{w}_m^\mathsf{T}\boldsymbol{d}_t) \tag{4.2}$$

and randomly generated $\boldsymbol{W} \triangleq [\boldsymbol{w}_1, \dots, \boldsymbol{w}_M]^\mathsf{T} \in \mathbb{R}^{M\times N}$. Note that y_m in (4.2) can be interpreted as a sample of the empirical characteristic function [58], i.e.,

$$\phi(\boldsymbol{w}_m) = \int_{\mathbb{R}^N} p(\boldsymbol{d}) \exp(\mathrm{j}\boldsymbol{w}_m^\mathsf{T}\boldsymbol{d})\, \mathrm{d}\boldsymbol{d} \tag{4.3}$$

under the empirical distribution $p(\boldsymbol{d}) = \frac{1}{T}\sum_{t=1}^{T}\delta(\boldsymbol{d} - \boldsymbol{d}_t)$, with Dirac $\delta(\cdot)$. Here, each $\boldsymbol{w}_m$ can be interpreted as a multidimensional frequency sample. The process of sketching $\boldsymbol{D}$ down to $\boldsymbol{y}$ via (4.2) costs $O(TMN)$ operations, but it can be performed efficiently in an online and/or distributed manner.

To recover the centroids $\boldsymbol{X}$ from $\boldsymbol{y}$, the state-of-the-art algorithm is *compressed learning via orthogonal matching pursuit with replacement* (CL-OMPR) [55, 56]. It aims to solve

$$\arg\min_{\boldsymbol{X}} \min_{\boldsymbol{\alpha}:\mathbf{1}^{\mathsf{T}}\boldsymbol{\alpha}=1} \sum_{m=1}^{M} \left| y_m - \sum_{k=1}^{K} \alpha_k \exp(\mathrm{j}\boldsymbol{w}_m^{\mathsf{T}}\boldsymbol{x}_k) \right|^2 \tag{4.4}$$

using a greedy heuristic inspired by the *orthogonal matching pursuit* (OMP) algorithm [59] popular in compressed sensing. With sketch length $M \geq 10KN$, CL-OMPR typically recovers centroids of similar or better quality to those attained with k-means++. One may wonder, however, whether it is possible to recover accurate centroids with sketch lengths closer to the counting bound $M = KN$. Also, since CL-OMPR's computational complexity is $O(MNK^2)$, one may wonder whether it is possible to recover accurate centroids with computational complexity $O(MNK)$.

4.1.2 Contributions

To recover the centroids $\boldsymbol{X}$ from a sketch $\boldsymbol{y}$ of the form in (4.2), we propose the *compressive learning via approximate message passing* (CL-AMP) algorithm, with computational complexity $O(MNK)$. Numerical experiments show that CL-AMP accurately recovers centroids from sketches of length $M = 2KN$ in most cases, which is an improvement over CL-OMPR. Experiments also show that CL-AMP recovers centroids faster and more accurately than k-means++ in certain operating regimes, such as when T is large.

The remainder of this chapter is organized as follows. In Section 4.2, we derive CL-AMP after reviewing relevant background on approximate message passing (AMP) algorithms. In Section 4.3, we present numerical experiments using synthetic and MNIST data, and we apply CL-AMP to multidimensional frequency estimation. In Section 4.4, we conclude.

4.2 Compressive Learning via AMP

4.2.1 High-Dimensional Inference Framework

CL-AMP treats centroid recovery as a high-dimensional inference problem rather than an optimization problem like minimizing (4.1) or (4.4). In particular, it models the data using a Gaussian mixture model (GMM)

$$\boldsymbol{d}_t \sim \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{x}_k, \boldsymbol{\Phi}_k), \tag{4.5}$$

where the centroids $\boldsymbol{x}_k$ act as the GMM means, and the GMM weights α_k and covariances $\boldsymbol{\Phi}_k$ are treated as unknown parameters. To recover the centroids $\boldsymbol{X} \triangleq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K]$ from $\boldsymbol{y}$, CL-AMP computes an approximation to the MMSE estimate

$$\widehat{\boldsymbol{X}} = \mathrm{E}\{\boldsymbol{X} \,|\, \boldsymbol{y}\}, \tag{4.6}$$

where the expectation is taken over the posterior density

$$p(\boldsymbol{X}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{X}) p(\boldsymbol{X}). \tag{4.7}$$

In (4.7), $p(\boldsymbol{y}|\boldsymbol{X})$ is the likelihood function of $\boldsymbol{X}$, and $p(\boldsymbol{X})$ is the prior density on $\boldsymbol{X}$. The dependence of $p(\boldsymbol{y}|\boldsymbol{X})$ on $\{\alpha_k\}$ and $\{\boldsymbol{\Phi}_k\}$ will be detailed in the sequel.

The form of the sketch in (4.2) implies that, when conditioning on the centroids $\boldsymbol{X}$ (and the frequencies $\boldsymbol{W}$), the elements of $\boldsymbol{y}$ can be modeled as i.i.d. In other words, the sketch $\boldsymbol{y}$ follows a generalized linear model (GLM) [60]. To make this precise, let us define the normalized frequency vectors

$$\boldsymbol{a}_m \triangleq \boldsymbol{w}_m / g_m \text{ with } g_m \triangleq \|\boldsymbol{w}_m\| \tag{4.8}$$

and the (normalized) transform outputs

$$\boldsymbol{z}_m^{\mathsf{T}} \triangleq \boldsymbol{a}_m^{\mathsf{T}} \boldsymbol{X} \in \mathbb{R}^K. \tag{4.9}$$

Then $p(\boldsymbol{y}|\boldsymbol{X})$ takes the form of a GLM, i.e.,

$$p(\boldsymbol{y}|\boldsymbol{X}) = \prod_{m=1}^{M} p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{a}_m^{\mathsf{T}}\boldsymbol{X}), \tag{4.10}$$

for a conditional pdf $p_{\mathsf{y}|\mathbf{z}}$ that will be detailed in the sequel.

From (4.2) and the definitions of $\boldsymbol{a}_m$ and g_m in (4.8), we have

$$\begin{aligned} y_m &= \frac{1}{T}\sum_{t=1}^{T} \exp(\mathrm{j}\boldsymbol{w}_m^{\mathsf{T}}\boldsymbol{d}_t) && (4.11)\\ &\approx \mathrm{E}\{\exp(\mathrm{j}\boldsymbol{w}_m^{\mathsf{T}}\boldsymbol{d}_t)\} && (4.12)\\ &= \sum_{k=1}^{K} \alpha_k \exp\left(\mathrm{j}g_m \underbrace{\boldsymbol{a}_m^{\mathsf{T}}\boldsymbol{x}_k}_{\triangleq\, z_{mk}} - \frac{g_m^2}{2}\underbrace{\boldsymbol{a}_m^{\mathsf{T}}\boldsymbol{\Phi}_k\boldsymbol{a}_m}_{\triangleq\, \tau_{mk}}\right), && (4.13)\end{aligned}$$

where (4.12) holds under large T and (4.13) follows from the facts

$$\boldsymbol{w}_m^{\mathsf{T}}\boldsymbol{d}_t \sim \sum_{k=1}^{K} \alpha_k \mathcal{N}(g_m z_{mk}, g_m^2\tau_{mk}) \tag{4.14}$$

under (4.5) and the following well-known result [61, p.153]

$$\mathrm{E}\{e^{\mathrm{j}x}\} = \exp\left(\mathrm{j}\mu - \sigma^2/2\right) \text{ when } x \sim \mathcal{N}(\mu, \sigma^2). \tag{4.15}$$

For $\boldsymbol{a}_m$ distributed uniformly on the sphere, the elements $\{\tau_{mk}\}_{m=1}^{M}$ in (4.13) concentrate as $N \to \infty$ [62], in that

$$\tau_{mk} \xrightarrow{p} \mathrm{E}\{\tau_{mk}\} = \mathrm{tr}(\boldsymbol{\Phi}_k)/N \triangleq \tau_k, \tag{4.16}$$

as long as the peak-to-average eigenvalue ratio of $\boldsymbol{\Phi}_k$ remains bounded. Thus, for large T and N, (4.13) and (4.16) imply that

$$y_m = \sum_{k=1}^{K} \alpha_k \exp\left(\mathrm{j}g_m z_{mk} - \frac{g_m^2\tau_k}{2}\right), \tag{4.17}$$

which can be rephrased as

$$p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}_m; \boldsymbol{\alpha}, \boldsymbol{\tau}) = \delta\left(y_m - \sum_{k=1}^{K} \alpha_k \exp\left(\mathrm{j}g_m z_{mk} - \frac{g_m^2\tau_k}{2}\right)\right) \tag{4.18}$$

where $\boldsymbol{\tau} \triangleq [\tau_1, \ldots, \tau_K]^{\mathsf{T}}$ and $\boldsymbol{\alpha} \triangleq [\alpha_1, \ldots, \alpha_K]^{\mathsf{T}}$ are hyperparameters of the GLM that will be estimated from $\boldsymbol{y}$.

For the CL-AMP framework, any prior of the form

$$p(\boldsymbol{X}) = \prod_{n=1}^{N} p_{\mathsf{x}}(\boldsymbol{x}_n^{\mathsf{T}}) \tag{4.19}$$

is admissible, where (with some abuse of notation) $\boldsymbol{x}_n^{\mathsf{T}}$ denotes the nth row of $\boldsymbol{X}$. For all experiments in Section 4.3, we used the trivial prior $p(\boldsymbol{X}) \propto 1$.

In summary, CL-AMP aims to compute the MMSE estimate of $\boldsymbol{X} \in \mathbb{R}^{N\times K}$ from the sketch $\boldsymbol{y} \in \mathbb{C}^M$ under the prior $\boldsymbol{X} \sim \prod_{n=1}^{N} p_{\mathsf{x}}(\boldsymbol{x}_n)$ from (4.19) and the likelihood $\boldsymbol{y} \sim \prod_{m=1}^{M} p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}_m; \boldsymbol{\alpha}, \boldsymbol{\tau})$ from (4.18), where $\boldsymbol{z}_m^{\mathsf{T}}$ is the mth row of $\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{X} \in \mathbb{R}^{M\times K}$ and $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ is a large random matrix with rows $\{\boldsymbol{a}_m^{\mathsf{T}}\}$ distributed uniformly on the unit sphere. CL-AMP estimates the values of $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$ from the sketch prior to estimating $\boldsymbol{X}$, as detailed in the sequel.

4.2.2 Approximate Message Passing

Exactly computing the MMSE estimate of $\boldsymbol{X}$ from $\boldsymbol{y}$ is impractical due to the form of $p_{\mathsf{y}|\mathbf{z}}$. Instead, one might consider approximate inference via the sum-product algorithm (SPA), but even the SPA is intractable due to the form of $p_{\mathsf{y}|\mathbf{z}}$. Given the presence of a large random matrix $\boldsymbol{A}$ in the problem formulation, we instead leverage *approximate message passing* (AMP) methods. In particular, we propose to apply the *simplified hybrid generalized AMP* (SHyGAMP) methodology from Chapters 2 and 3 (as well as [9]), while simultaneously estimating $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$ through expectation maximization (EM). A brief background on AMP methods will now be provided to justify our approach.

The original AMP algorithm of Donoho, Maleki, and Montanari [1] was designed to estimate i.i.d. $\boldsymbol{x}$ under the standard linear model (i.e., $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}$ with known $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ and additive white Gaussian noise $\boldsymbol{n}$). The generalized AMP (GAMP) algorithm of Rangan [2] extended AMP to the generalized linear model (i.e., $\boldsymbol{y} \sim p(\boldsymbol{y}|\boldsymbol{z})$ for $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}$ and separable $p(\boldsymbol{y}|\boldsymbol{z}) = \prod_{m=1}^{M} p(y_m|z_m)$). Both AMP and GAMP give accurate approximations of the SPA under large i.i.d. sub-Gaussian $\boldsymbol{A}$, while maintaining a computational complexity of only $O(MN)$. Furthermore, both can be rigorously analyzed via the state-evolution framework, which proves that they compute MMSE optimal estimates of $\boldsymbol{x}$ in certain regimes [3].

A limitation of AMP [1] and GAMP [2] is that they treat only problems with i.i.d. estimand $\boldsymbol{x}$ and separable likelihood $p(\boldsymbol{y}|\boldsymbol{z}) = \prod_{m=1}^{M} p(y_m|z_m)$. Thus, *Hybrid GAMP* (HyGAMP) [5] was developed to tackle problems with a structured prior and/or likelihood. HyGAMP could be applied to the compressive learning problem described in Section 4.2.1, but it would require computing and inverting $O(N+M)$ covariance matrices of dimension K at each iteration. For this reason, we instead apply the *simplified HyGAMP* (SHyGAMP) algorithm from Section 2.3 [9], which uses diagonal covariance matrices in HyGAMP to reduce its computational complexity. As described in [9], SHyGAMP can be readily combined with the EM algorithm to learn the hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$.

4.2.3 From SHyGAMP to CL-AMP

The SHyGAMP algorithm was described in detail in Chapters 2 and 3. Table 4.1 summarizes the SPA-SHyGAMP algorithm using the language of Section 4.2.1 (see Table 2.1 in Chapter 2 for the general HyGAMP algorithm).

The SHyGAMP algorithm can be applied to many different problems via appropriate choice of $p_{\mathsf{y}|\mathsf{z}}$ and p_{x}. To apply SHyGAMP to sketched clustering, we choose $p_{\mathsf{y}|\mathsf{z}}$ and p_{x} as described in Section 4.2.1, which yield approximate posterior distributions

$$p_{\mathsf{x}|\mathsf{r}}(\boldsymbol{x}_n|\widehat{\boldsymbol{r}}_n;\boldsymbol{Q}^{\mathsf{r}}) = \frac{p_{\mathsf{x}}(\boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}^{\mathsf{r}})}{\int p_{\mathsf{x}}(\boldsymbol{x}'_n)\mathcal{N}(\boldsymbol{x}'_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}^{\mathsf{r}})\,\mathrm{d}\boldsymbol{x}'_n}. \tag{4.20}$$

and

$$\begin{aligned} &p_{\mathsf{z}|\mathsf{y},\mathsf{p}}(\boldsymbol{z}_m|y_m,\widehat{\boldsymbol{p}}_m;\boldsymbol{Q}^{\mathsf{p}},\boldsymbol{\alpha},\boldsymbol{\tau}) \\ &= \frac{p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m;\boldsymbol{\alpha},\boldsymbol{\tau})\mathcal{N}(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}})}{\int p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}'_m;\boldsymbol{\alpha},\boldsymbol{\tau})\mathcal{N}(\boldsymbol{z}'_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}})\,\mathrm{d}\boldsymbol{z}'_m}. \end{aligned} \tag{4.21}$$

As we will see, the main challenge is evaluating lines 4-5 of Table 4.1 for the $p_{\mathsf{y}|\mathsf{z}}$ in (4.18).

Inference of $\boldsymbol{z}_m$

For lines 4-5 of Table 4.1, we would like to compute the mean and variance

$$\widehat{z}_{mk} = \frac{\int_{\mathbb{R}^K} z_{mk} p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)\mathcal{N}\Big(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}\Big)\,\mathrm{d}\boldsymbol{z}_m}{C_m} \tag{4.22}$$

$$q^{\mathsf{z}}_{mk} = \frac{\int_{\mathbb{R}^K} (z_{mk}-\widehat{z}_{mk})^2 p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)\mathcal{N}\Big(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}\Big)\,\mathrm{d}\boldsymbol{z}_m}{C_m}, \tag{4.23}$$

where q^{z}_{mk} is the kth element of $\boldsymbol{q}^{\mathsf{z}}_m$ and

$$C_m = \int_{\mathbb{R}^K} p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{z}_m)\mathcal{N}\Big(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}\Big)\,\mathrm{d}\boldsymbol{z}_m. \tag{4.24}$$

However, due to the form of $p_{\mathsf{y}|\mathsf{z}}$ in (4.18), we are not able to find closed-form expressions for $\widehat{z}_{mk}$ or q^{z}_{mk}. Thus, we propose to approximate $\widehat{z}_{mk}$ and q^{z}_{mk} as follows. The main idea behind our approximation is to write (4.17) as

$$\begin{aligned} y_m &= \alpha_k \exp(-g_m^2\tau_k/2)\exp\Big(\mathrm{j}g_m z_{mk}\Big) \\ &\quad + \sum_{l\neq k} \alpha_l \exp(-g_m^2\tau_l/2)\exp\Big(\mathrm{j}g_m(z_{ml})\Big) \end{aligned} \tag{4.25}$$

Require: Measurements $\boldsymbol{y} \in \mathbb{C}^M$, matrix $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ with $\|\boldsymbol{A}\|_F^2 = M$, pdfs $p_{\mathsf{x}|\mathsf{r}}(\cdot|\cdot)$ and $p_{\mathsf{z}|\mathsf{y},\mathsf{p}}(\cdot|\cdot,\cdot;\boldsymbol{\alpha},\boldsymbol{\tau})$ from (4.20) and (4.21), initial $\widehat{\boldsymbol{X}}_0 \in \mathbb{R}^{N\times K}$ and $q^{\mathsf{p}} = q_0^{\mathsf{p}} \in \mathbb{R}_+^K$.

```
1: $\widehat{\boldsymbol{S}} \leftarrow \boldsymbol{0}$, $\widehat{\boldsymbol{X}} \leftarrow \widehat{\boldsymbol{X}}_0$.
2: repeat
3:     $\widehat{\boldsymbol{P}} \leftarrow \boldsymbol{A}\widehat{\boldsymbol{X}} - \widehat{\boldsymbol{S}}\,\mathrm{diag}(q^{\mathsf{p}})$
4:     $q_m^{\mathsf{z}} \leftarrow \mathrm{diag}\Big(\mathrm{Cov}\Big\{\boldsymbol{z}_m \Big| y_m, \widehat{\boldsymbol{p}}_m; \mathrm{diag}(q^{\mathsf{p}}), \boldsymbol{\alpha}, \boldsymbol{\tau}\Big\}\Big),\ m = 1...M$
5:     $\widehat{\boldsymbol{z}}_m \leftarrow \mathrm{E}\Big\{\boldsymbol{z}_m \Big| y_m, \widehat{\boldsymbol{p}}_m; \mathrm{diag}(q^{\mathsf{p}}), \boldsymbol{\alpha}, \boldsymbol{\tau}\Big\},\ m = 1...M$
6:     $q^{\mathsf{s}} \leftarrow \boldsymbol{1} \oslash q^{\mathsf{p}} - \Big(\frac{1}{M}\sum_{m=1}^M q_m^{\mathsf{z}}\Big) \oslash (q^{\mathsf{p}} \odot q^{\mathsf{p}})$
7:     $\widehat{\boldsymbol{S}} \leftarrow (\widehat{\boldsymbol{Z}} - \widehat{\boldsymbol{P}})\,\mathrm{diag}(q^{\mathsf{p}})^{-1}$
8:     $q^{\mathsf{r}} \leftarrow \frac{N}{M}\boldsymbol{1} \oslash q^{\mathsf{s}}$
9:     $\widehat{\boldsymbol{R}} \leftarrow \widehat{\boldsymbol{X}} + \boldsymbol{A}^{\mathsf{T}}\widehat{\boldsymbol{S}}\,\mathrm{diag}(q^{\mathsf{r}})$
10:    $q_n^{\mathsf{x}} \leftarrow \mathrm{diag}\Big(\mathrm{Cov}\Big\{\boldsymbol{x}_n \Big| \widehat{\boldsymbol{r}}_n; \mathrm{diag}(q^{\mathsf{r}})\Big\}\Big),\ n = 1...N$
11:    $\widehat{\boldsymbol{x}}_n \leftarrow \mathrm{E}\Big\{\boldsymbol{x}_n \Big| \widehat{\boldsymbol{r}}_n; \mathrm{diag}(q^{\mathsf{r}})\Big\},\ n = 1...N$
12:    $q^{\mathsf{p}} \leftarrow \frac{1}{N}\sum_{n=1}^N q_n^{\mathsf{x}}$
13: until convergence
14: return $\widehat{\boldsymbol{X}}$
```

Table 4.1: SPA-SHyGAMP for Sketched Clustering

and treat the sum over l as complex Gaussian. For the remainder of this section, we suppress the subscripts "m" and "$\mathsf{y}|\mathsf{z}$" to simplify the notation.

We begin by writing (4.25) as

$$\begin{aligned} y = &\underbrace{\alpha_k \exp(-g^2\tau_k/2)}_{\triangleq\, \beta_k} \exp\Big(\mathrm{j}\,\underbrace{g(z_k + n_k)}_{\triangleq\, \theta_k}\Big) \\ &+ \sum_{l\neq k} \underbrace{\alpha_l \exp(-g^2\tau_l/2)}_{=\, \beta_l} \underbrace{\exp\Big(\mathrm{j}g(z_l + n_l)\Big)}_{\triangleq\, v_l}, \end{aligned} \tag{4.26}$$

where we introduced i.i.d. $n_k \sim \mathcal{N}(0, q^{\mathsf{n}})$ to facilitate the derivation in the sequel. Eventually we will take $q^{\mathsf{n}} \to 0$, in which case (4.26) exactly matches (4.25).

Next we derive an expression for the marginal posterior $p(z_k|y)$ under the pseudo-prior $z_k \sim \mathcal{N}(\widehat{p}_k, q_k^{\mathsf{p}})\ \forall k$. First,

$$\begin{aligned}
p(z_k|y) &= \int_{\mathbb{R}^K} p(\boldsymbol{z}, \theta_k|y)\,\mathrm{d}\theta_k\,\mathrm{d}\boldsymbol{z}_{\backslash k} && (4.27)\\
&= \frac{1}{p(y)} \int_{\mathbb{R}^K} p(y|\boldsymbol{z}, \theta_k) p(\theta_k|\boldsymbol{z}) p(\boldsymbol{z})\,\mathrm{d}\theta_k\,\mathrm{d}\boldsymbol{z}_{\backslash k} && (4.28)\\
&= \frac{1}{p(y)} \int_{\mathbb{R}^K} p(y|\boldsymbol{z}_{\backslash k}, \theta_k) \mathcal{N}(\theta_k; g z_k, g^2 q^{\mathsf{n}})\\
&\qquad \times \prod_{l=1}^{K} \mathcal{N}(z_l; \widehat{p}_l, q_l^{\mathsf{p}})\,\mathrm{d}\theta_k\,\mathrm{d}\boldsymbol{z}_{\backslash k}, && (4.29)
\end{aligned}$$

where $\boldsymbol{z}_{\backslash k} \triangleq [z_1, \dots, z_{k-1}, z_{k+1}, \dots, z_K]^{\mathsf{T}}$. A change-of-variables from z_l to $\widetilde{z}_l \triangleq z_l - \widehat{p}_l$ for all $l \neq k$ gives

$$\begin{aligned}
p(z_k|y) = \frac{\mathcal{N}(z_k; \widehat{p}_k, q_k^{\mathsf{p}})}{p(y)} & \int_{\mathbb{R}} \mathcal{N}(\theta_k; g z_k, g^2 q^{\mathsf{n}}) && (4.30)\\
& \times \left[\int_{\mathbb{R}^{K-1}} p(y|\widetilde{\boldsymbol{z}}_{\backslash k}, \theta_k) \prod_{l \neq k} \mathcal{N}(\widetilde{z}_l; 0, q_l^{\mathsf{p}})\,\mathrm{d}\widetilde{\boldsymbol{z}}_{\backslash k} \right] \mathrm{d}\theta_k,
\end{aligned}$$

where $p(y|\widetilde{\boldsymbol{z}}_{\backslash k}, \theta_k)$ is associated with the generative model

$$y = \beta_k \exp(\mathrm{j}\theta_k) + \sum_{l \neq k} \beta_l \exp\Big(\mathrm{j} g(\widehat{p}_l + \widetilde{z}_l + n_l)\Big) \tag{4.31}$$

with i.i.d. $n_l \sim \mathcal{N}(0, q^{\mathsf{n}})$. Now, because $\widetilde{z}_l$ and n_l are (apriori) mutually independent zero-mean Gaussian variables, we can work directly with the sum $\widetilde{n}_l \triangleq \widetilde{z}_l + n_l \sim \mathcal{N}(0, q_l^{\mathsf{p}} + q^{\mathsf{n}})$ and thus bypass the inner integral in (4.30). This allows us to write

$$p(z_k|y) = \frac{\mathcal{N}(z_k; \widehat{p}_k, q_k^{\mathsf{p}})}{p(y)} \int_{\mathbb{R}} \mathcal{N}(\theta_k; g z_k, g^2 q^{\mathsf{n}}) p(y|\theta_k)\,\mathrm{d}\theta_k, \tag{4.32}$$

where $p(y|\theta_k)$ is associated with the generative model

$$y = \beta_k \exp(\mathrm{j}\theta_k) + \sum_{l \neq k} \beta_l \underbrace{\exp(\mathrm{j} g(\widehat{p}_l + \widetilde{n}_l))}_{= v_l} \tag{4.33}$$

with i.i.d. $\widetilde{n}_l \sim \mathcal{N}(0, q^{\mathsf{p}}_l + q^{\mathsf{n}})$. Recalling that $y \in \mathbb{C}$, it will sometimes be useful to write (4.33) as

$$\begin{aligned}\begin{bmatrix}\mathrm{Re}\{y\}\\ \mathrm{Im}\{y\}\end{bmatrix} \sim \mathcal{N}\Bigg(\beta_k \begin{bmatrix}\cos(\theta_k)\\ \sin(\theta_k)\end{bmatrix} + \sum_{l\neq k} \beta_l \,\mathrm{E}\left\{ \begin{bmatrix}\mathrm{Re}\{v_l\}\\ \mathrm{Im}\{v_l\}\end{bmatrix} \right\},\\ \sum_{l\neq k} \beta_l^2 \,\mathrm{Cov}\left\{ \begin{bmatrix}\mathrm{Re}\{v_l\}\\ \mathrm{Im}\{v_l\}\end{bmatrix} \right\}\Bigg).\end{aligned} \tag{4.34}$$

To compute the posterior mean of z_k, (4.32) implies

$$\widehat{z}_k \triangleq \mathrm{E}\{z_k|y\} = \int_{\mathbb{R}} z_k \, p(z_k|y) \,\mathrm{d}z_k \tag{4.35}$$

$$= \frac{1}{p(y)} \int_{\mathbb{R}} \left[\int_{\mathbb{R}} z_k \,\mathcal{N}(gz_k; \theta_k, g^2 q^{\mathsf{n}}) \mathcal{N}(z_k; \widehat{p}_k, q^{\mathsf{p}}_k) \,\mathrm{d}z_k \right] \times p(y|\theta_k) \,\mathrm{d}\theta_k \tag{4.36}$$

$$= \int_{\mathbb{R}} \left[\int_{\mathbb{R}} z_k \,\mathcal{N}\left(z_k; \frac{\frac{\theta_k/g}{q^{\mathsf{n}}} + \frac{\widehat{p}_k}{q^{\mathsf{p}}_k}}{\frac{1}{q^{\mathsf{n}}} + \frac{1}{q^{\mathsf{p}}_k}}, \frac{1}{\frac{1}{q^{\mathsf{n}}} + \frac{1}{q^{\mathsf{p}}_k}} \right) \mathrm{d}z_k \right] \times \underbrace{\frac{\mathcal{N}\Big(\theta_k; g\widehat{p}_k, g^2(q^{\mathsf{n}} + q^{\mathsf{p}}_k)\Big)\, p(y|\theta_k)}{p(y)}}_{= p(\theta_k|y)} \mathrm{d}\theta_k \tag{4.37}$$

$$= \int_{\mathbb{R}} \frac{\frac{\theta_k/g}{q^{\mathsf{n}}} + \frac{\widehat{p}_k}{q^{\mathsf{p}}_k}}{\frac{1}{q^{\mathsf{n}}} + \frac{1}{q^{\mathsf{p}}_k}} p(\theta_k|y) \,\mathrm{d}\theta_k \tag{4.38}$$

$$= \frac{\widehat{p}_k}{q^{\mathsf{p}}_k/q^{\mathsf{n}} + 1} + \frac{\widehat{\theta}_k/g}{1 + q^{\mathsf{n}}/q^{\mathsf{p}}_k} \quad \text{for } \widehat{\theta}_k \triangleq \int_{\mathbb{R}} \theta_k \, p(\theta_k|y) \,\mathrm{d}\theta_k, \tag{4.39}$$

where the Gaussian pdf multiplication rule[14] was used in (4.37) and where $\widehat{\theta}_k$ denotes the posterior mean of θ_k.

[14] $\mathcal{N}(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{A})\mathcal{N}(\boldsymbol{x}; \boldsymbol{b}, \boldsymbol{B}) = \mathcal{N}(\boldsymbol{0}; \boldsymbol{a} - \boldsymbol{b}, \boldsymbol{A} + \boldsymbol{B})\mathcal{N}\big(\boldsymbol{x}; (\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1}(\boldsymbol{A}^{-1}\boldsymbol{a} + \boldsymbol{B}^{-1}\boldsymbol{b}), (\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1}\big)$.

For the posterior variance of z_k, a similar approach gives

$$q^{\mathsf{z}}_k \triangleq \mathrm{var}\{z_k|y\} = \int_{\mathbb{R}} \Big(z_k - \widehat{z}_k\Big)^2 p(z_k|y)\,\mathrm{d}z_k \tag{4.40}$$

$$= \frac{1}{p(y)} \int_{\mathbb{R}} \bigg[\int_{\mathbb{R}} (z_k - \widehat{z}_k)^2 \,\mathcal{N}(gz_k; \theta_k, g^2 q^{\mathsf{n}}) \times \mathcal{N}(z_k; \widehat{p}_k, q^{\mathsf{p}}_k)\,\mathrm{d}z_k \bigg] p(y|\theta_k)\,\mathrm{d}\theta_k \tag{4.41}$$

$$= \int_{\mathbb{R}} \left[\int_{\mathbb{R}} (z_k - \widehat{z}_k)^2 \,\mathcal{N}\left(z_k; \frac{\frac{\theta_k/g}{q^{\mathsf{n}}} + \frac{\widehat{p}_k}{q^{\mathsf{p}}_k}}{\frac{1}{q^{\mathsf{n}}} + \frac{1}{q^{\mathsf{p}}_k}}, \frac{1}{\frac{1}{q^{\mathsf{n}}} + \frac{1}{q^{\mathsf{p}}_k}} \right) \mathrm{d}z_k \right] \times p(\theta_k|y)\,\mathrm{d}\theta_k. \tag{4.42}$$

Using a change-of-variables from z_k to $\widetilde{z}_k \triangleq z_k - \widehat{z}_k$, we get

$$q^{\mathsf{z}}_k = \int_{\mathbb{R}} \left[\int_{\mathbb{R}} \widetilde{z}_k^2 \,\mathcal{N}\left(\widetilde{z}_k; \frac{\frac{\theta_k/g}{q^{\mathsf{n}}} - \frac{\widehat{\theta}_k/g}{q^{\mathsf{n}}}}{\frac{1}{q^{\mathsf{n}}} + \frac{1}{q^{\mathsf{p}}_k}}, \frac{1}{\frac{1}{q^{\mathsf{n}}} + \frac{1}{q^{\mathsf{p}}_k}} \right) \mathrm{d}\widetilde{z}_k \right] \times p(\theta_k|y)\,\mathrm{d}\theta_k \tag{4.43}$$

$$= \int_{\mathbb{R}} \left[\left(\frac{(\theta_k - \widehat{\theta}_k)/g}{1 + q^{\mathsf{n}}/q^{\mathsf{p}}_k} \right)^2 + \frac{q^{\mathsf{n}}}{1 + q^{\mathsf{n}}/q^{\mathsf{p}}_k} \right] p(\theta_k|y)\,\mathrm{d}\theta_k \tag{4.44}$$

$$= \frac{q^{\mathsf{n}}}{1 + q^{\mathsf{n}}/q^{\mathsf{p}}_k} + \frac{1}{g^2} \left(\frac{1}{1 + q^{\mathsf{n}}/q^{\mathsf{p}}_k} \right)^2 \underbrace{\int_{\mathbb{R}} (\theta_k - \widehat{\theta}_k)^2 \, p(\theta_k|y)\,\mathrm{d}\theta_k}_{\triangleq\, q^{\theta}_k \,=\, \mathrm{var}\{\theta_k|y\}}. \tag{4.45}$$

The computation of $\widehat{z}_k$ and q^{z}_k is still complicated by the form of the posterior $p(\theta_k|y)$ implied by (4.33). To circumvent this problem, we propose to apply a Gaussian approximation to the sum in (4.33). Because $\{\widetilde{n}_l\}_{\forall l \neq k}$ are mutually independent, the mean and covariance of the sum in (4.33) are simply the sum of the means and covariances (respectively) of the $K-1$ terms making up the sum. Recalling (4.34), this implies that

$$p\left(\begin{bmatrix} \mathrm{Re}\{y\} \\ \mathrm{Im}\{y\} \end{bmatrix} \middle| \theta_k \right) \approx \mathcal{N}\left(\begin{bmatrix} \mathrm{Re}\{y\} \\ \mathrm{Im}\{y\} \end{bmatrix}; \beta_k \begin{bmatrix} \cos(\theta_k) \\ \sin(\theta_k) \end{bmatrix} + \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \right) \tag{4.46}$$

with

$$\boldsymbol{\mu}_k = \sum_{l\neq k} \alpha_l e^{-g^2(\tau_k+q_k^{\mathsf{p}})/2} \begin{bmatrix}\cos(g\widehat{p}_l)\\ \sin(g\widehat{p}_l)\end{bmatrix} \tag{4.47}$$

$$\begin{aligned}\boldsymbol{\Sigma}_k &= \frac{1}{2}\sum_{l\neq k}\beta_l^2\Big(1-e^{-g^2 q_l^{\mathsf{p}}}\Big)\\ &\quad\times\left(\boldsymbol{I}-e^{-g^2 q_l^{\mathsf{p}}}\begin{bmatrix}\cos(2g\widehat{p}_l) & \sin(2g\widehat{p}_l)\\ \sin(2g\widehat{p}_l) & -\cos(2g\widehat{p}_l)\end{bmatrix}\right).\end{aligned} \tag{4.48}$$

We note that (4.47) and (4.48) were obtained using

$$\mathrm{E}\Big\{\mathrm{Re}\{v_l\}\Big\} = \exp\Big(-g^2 q_l^{\mathsf{p}}/2\Big)\cos(g\widehat{p}_l) \tag{4.49}$$

$$\mathrm{E}\Big\{\mathrm{Im}\{v_l\}\Big\} = \exp\Big(-g^2 q_l^{\mathsf{p}}/2\Big)\sin(g\widehat{p}_l) \tag{4.50}$$

$$2\,\mathrm{E}\Big\{\mathrm{Re}\{v_l\}^2\Big\} = 1+\exp\Big(-g^2 q_l^{\mathsf{p}}\Big)\cos(2g\widehat{p}_l) \tag{4.51}$$

$$2\,\mathrm{E}\Big\{\mathrm{Im}\{v_l\}^2\Big\} = 1-\exp\Big(-g^2 q_l^{\mathsf{p}}\Big)\cos(2g\widehat{p}_l) \tag{4.52}$$

$$2\,\mathrm{E}\Big\{\mathrm{Re}\{v_l\}\,\mathrm{Im}\{v_l\}\Big\} = \exp\Big(-g^2 q_l^{\mathsf{p}}\Big)\sin(2g\widehat{p}_l), \tag{4.53}$$

which use the fact that, after letting $q^{\mathsf{n}} \to 0$,

$$\begin{aligned}\mathrm{E}\{v_l\} &= \int_{\mathbb{R}} \mathcal{N}(z_l;\widehat{p}_l, q_l^{\mathsf{p}})\exp(\mathrm{j}gz_l)\,\mathrm{d}z_l & (4.54)\\ &= \exp\Big(\mathrm{j}g\widehat{p}_l - g^2 q_l^{\mathsf{p}}/2\Big). & (4.55)\end{aligned}$$

Rewriting (4.46) as

$$\begin{aligned}&p\left(\beta_k^{-1}\begin{bmatrix}\mathrm{Re}\{y\}\\ \mathrm{Im}\{y\}\end{bmatrix}\middle|\theta_k\right)\\ &\quad\approx\mathcal{N}\left(\begin{bmatrix}\cos(\theta_k)\\ \sin(\theta_k)\end{bmatrix};\beta_k^{-1}\begin{bmatrix}\mathrm{Re}\{y\}\\ \mathrm{Im}\{y\}\end{bmatrix}-\beta_k^{-1}\boldsymbol{\mu}_k,\beta_k^{-2}\boldsymbol{\Sigma}_k\right),\end{aligned} \tag{4.56}$$

the right side of (4.56) can be recognized as being proportional to the generalized von Mises (GvM) density over $\theta_k \in [0, 2\pi)$ from [63]. Under this GvM approximation, we have [63] that

$$p(y|\theta_k) \propto \exp\Big(\kappa_k\cos(\theta_k-\zeta_k)+\overline{\kappa}_k\cos[2(\theta_k-\overline{\zeta}_k)]\Big) \tag{4.57}$$

for parameters $\kappa_k, \overline{\kappa}_k > 0$ and $\zeta_k, \overline{\zeta}_k \in [0, 2\pi)$ defined from $\beta_k^{-1} y$, $\beta_k^{-1}\boldsymbol{\mu}_k$, and $\beta_k^{-2}\boldsymbol{\Sigma}_k$. In particular,

$$\kappa_k \cos(\zeta_k) = -\frac{1}{1-\rho_k^2}\left(\frac{\rho_k \overline{\nu}_k}{\sigma_k \overline{\sigma}_k} - \frac{\nu_k}{\sigma_k^2}\right) \tag{4.58}$$

$$\kappa_k \sin(\zeta_k) = -\frac{1}{1-\rho_k^2}\left(\frac{\rho_k \nu_k}{\sigma_k \overline{\sigma}_k} - \frac{\overline{\nu}_k}{\overline{\sigma}_k^2}\right) \tag{4.59}$$

$$\overline{\kappa}_k \cos(2\overline{\zeta}_k) = -\frac{1}{4(1-\rho_k^2)}\left(\frac{1}{\sigma_k^2} - \frac{1}{\overline{\sigma}_k^2}\right) \tag{4.60}$$

$$\overline{\kappa}_k \sin(2\overline{\zeta}_k) = \frac{\rho_k}{2(1-\rho_k^2)\sigma_k \overline{\sigma}_k}, \tag{4.61}$$

where

$$\begin{bmatrix}\nu_k \\ \overline{\nu}_k\end{bmatrix} \triangleq \beta_k^{-1}\left(\begin{bmatrix}\mathrm{Re}\{y\} \\ \mathrm{Im}\{y\}\end{bmatrix} - \boldsymbol{\mu}_k\right) \tag{4.62}$$

$$\begin{bmatrix}\sigma_k^2 & \rho_k \sigma_k \overline{\sigma}_k \\ \rho_k \sigma_k \overline{\sigma}_k & \overline{\sigma}_k^2\end{bmatrix} \triangleq \beta_k^{-2}\boldsymbol{\Sigma}_k. \tag{4.63}$$

From (4.57) and the SHyGAMP pseudo-prior $z_k \sim \mathcal{N}(\widehat{p}_k, q_k^{\mathsf{p}})$, we see that the posterior on θ_k takes the form

$$p(\theta_k|y) \propto \mathcal{N}\Big(\theta_k; g\widehat{p}_k, g^2 q_k^{\mathsf{p}}\Big)\, p(y|\theta_k) \tag{4.64}$$

$$\propto \exp\left[\kappa_k \cos(\theta_k - \zeta_k) + \overline{\kappa}_k \cos[2(\theta_k - \overline{\zeta}_k)] - \frac{(\theta_k - g\widehat{p}_k)^2}{2g^2 q_k^{\mathsf{p}}}\right]. \tag{4.65}$$

We now face the task of computing $\widehat{\theta}_k = \mathrm{E}\{\theta_k|y\}$ and $q_k^{\theta} = \mathrm{var}\{\theta_k|y\}$ under (4.65). Since these quantities do not appear to be computable in closed form, we settle for an approximation, such as that based on the Laplace approximation [23] or numerical integration. For the Laplace approximation, we would first compute $\widehat{\theta}_{k,\mathsf{MAP}} \triangleq \arg\max_{\theta_k} \ln p(\theta_k|y)$ and then approximate $\widehat{\theta}_k \approx \widehat{\theta}_{k,\mathsf{MAP}}$ and $q_k^{\theta} \approx -\frac{\mathrm{d}^2}{\mathrm{d}\theta_k{}^2} \ln p(\theta_k|y)\Big|_{\theta_k = \widehat{\theta}_{k,\mathsf{MAP}}}$. However, since computing $\arg\max_{\theta_k} \ln p(\theta_k|y)$ is complicated due to the presence of multiple local maxima, we instead use numerical integration. For this, we suggest a grid of $N_{\text{pts}} N_{\text{per}} + 1$ uniformly-spaced points centered at $g\widehat{p}_k$ with width $2\pi N_{\text{per}}$, where $N_{\text{per}} = \left\lceil \frac{N_{\text{std}}}{\pi}\sqrt{g^2 q_k^{\mathsf{p}}} \right\rceil$. This choice of grid ensures that the sampling points cover at least N_{std} standard deviations of the prior on θ_k. We used $N_{\text{std}} = 4$ and $N_{\text{pts}} = 7$ in the numerical experiments in Section 4.3.

Finally, after approximating $\widehat{\theta}_k$ and q_k^{θ} via numerical integration, we set $\widehat{z}_k = \widehat{\theta}_k / g$ and $q_k^{\mathsf{z}} = q_k^{\theta}/g^2$.

Inference of $\boldsymbol{x}_n$

Recall that lines 10-11 of Table 4.1 support an arbitrary prior p_{x} on $\boldsymbol{x}_n$. For the experiments in Section 4.3, we used the trivial non-informative prior $p_{\mathsf{x}}(\boldsymbol{x}_n) \propto 1$, after which lines 10-11 reduce to

$$q_n^{\mathsf{x}} = q^{\mathsf{r}} \ \forall n \ \text{ and } \ \widehat{\boldsymbol{x}}_n = \widehat{\boldsymbol{r}}_n \ \forall n. \tag{4.66}$$

4.2.4 Initialization

We recommend initializing CL-AMP with $\widehat{\boldsymbol{X}} = \widehat{\boldsymbol{X}}_0$ and $q^{\mathsf{p}} = q_0^{\mathsf{p}}$, where $\widehat{\boldsymbol{X}}_0$ is drawn i.i.d. $\mathcal{N}(0, \sigma^2)$ and where $q_0^{\mathsf{p}} = \sigma^2 \mathbf{1}$, with σ^2 from (4.79) (as described in Section 4.2.7).

In some cases, running CL-AMP from $R > 1$ different random initializations can help avoid to spurious solutions. Here, CL-AMP is run from a different random initialization $\widehat{\boldsymbol{X}}_{0,r}$, for $r = 1, \dots, R$, and then the quality of the recovered solution $\widehat{\boldsymbol{X}}_r$ is evaluated by constructing the "estimated sketch" $\widehat{\boldsymbol{y}}_r$ via

$$\widehat{y}_{mr} = \sum_{k=1}^{K} \alpha_k \exp(-g_m^2 \tau_k) \exp(\mathrm{j} g_m \boldsymbol{a}_m^{\mathsf{T}} \widehat{\boldsymbol{x}}_{kr}) \tag{4.67}$$

recalling (4.9) and (4.17), and then measuring its distance to the true sketch $\boldsymbol{y}$. The initialization index is then selected as

$$r_* = \arg\min_r \|\boldsymbol{y} - \widehat{\boldsymbol{y}}_r\|, \tag{4.68}$$

and the centroids saved as $\widehat{\boldsymbol{X}} = \widehat{\boldsymbol{X}}_{r_*}$. In Section 4.3, we used $R = 2$ for all experiments.

4.2.5 Hyperparameter Tuning

The likelihood model $p_{\mathsf{y}|\mathbf{z}}$ in (4.18) depends on the unknown hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$. We propose to estimate these hyperparameters using a combination of *expectation maximization* (EM) and SHyGAMP, as suggested in [9] and detailed—for the simpler case of GAMP—in [4]. The idea is to run SHyGAMP using an estimate of $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$, update $\boldsymbol{\alpha}$ and $\boldsymbol{\tau}$ from the SHyGAMP outputs, and repeat until convergence. For the first estimate, we suggest to use $\alpha_k = \frac{1}{K}$ and $\tau_k = 0 \ \forall \ k$.

Extrapolating [4, eq. (23)] to the SHyGAMP case, the EM update of $(\boldsymbol{\alpha}, \boldsymbol{\tau})$ takes the form

$$\begin{aligned}(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\tau}}) = \underset{\boldsymbol{\alpha}\geq\mathbf{0}, \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{1}=1, \boldsymbol{\tau}>\mathbf{0}}{\arg\max} \sum_{m=1}^{M} \int_{\mathbb{R}^K} \mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{z}}_m, \boldsymbol{Q}^{\mathbf{z}}_m) \\ \times \ln p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}_m; \boldsymbol{\alpha}, \boldsymbol{\tau})\, \mathrm{d}\boldsymbol{z}_m,\end{aligned} \tag{4.69}$$

where $\widehat{\boldsymbol{z}}_m$ and $\boldsymbol{Q}^{\mathbf{z}}_m = \operatorname{diag}\{q^{\mathbf{z}}_m\}$ are obtained by running SHyGAMP to convergence under $(\boldsymbol{\alpha}, \boldsymbol{\tau})$. To proceed, we model the Dirac delta in (4.18) using a circular Gaussian pdf with vanishingly small variance $\epsilon > 0$, in which case

$$\begin{aligned}&\ln p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}_m; \boldsymbol{\alpha}, \boldsymbol{\tau}) \\ &= -\frac{1}{\epsilon}\left| y_m - \sum_{k=1}^{K} \alpha_k \exp\left(\mathrm{j} g_m z_{mk} - \frac{g_m^2 \tau_k}{2}\right)\right|^2 + \text{const.}\end{aligned} \tag{4.70}$$

Plugging (4.70) back into (4.69), we see that the constant and the $1/\epsilon$-scaling play no role in the optimization, and so we can discard them to obtain

$$(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\tau}}) = \underset{\boldsymbol{\alpha}\geq\mathbf{0}, \boldsymbol{\alpha}^{\mathsf{T}}\mathbf{1}=1, \boldsymbol{\tau}>\mathbf{0}}{\arg\min} \sum_{m=1}^{M} \int_{\mathbb{R}^K} \mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{z}}_m, \boldsymbol{Q}_m^{\mathsf{z}}) \times \left| y_m - \sum_{k=1}^{K} \alpha_k \exp\left(\mathrm{j} g_m z_{mk} - \frac{g_m^2 \tau_k}{2} \right) \right|^2 \mathrm{d}\boldsymbol{z}_m. \tag{4.71}$$

A closed-form solution to the optimization problem in (4.71) seems out of reach. Also, the optimization objective is convex in $\boldsymbol{\alpha}$ for fixed $\boldsymbol{\tau}$, and convex in $\boldsymbol{\tau}$ for fixed $\boldsymbol{\alpha}$, but not jointly convex in $[\boldsymbol{\alpha}^{\mathsf{T}}, \boldsymbol{\tau}^{\mathsf{T}}]$. Although the optimization problem (4.71) is difficult to solve, the solutions obtained by gradient projection (GP) [43] seem to work well in practice. Also, GP is made practical by closed-form gradient expressions. In particular, let

$$q_{mk} \triangleq \exp\left(-\frac{g_m^2 \tau_k}{2} \right) \tag{4.72}$$

$$\rho_{mk} \triangleq \exp\left(\mathrm{j} g_m \widehat{z}_{mk} - \frac{q_{mk}^{\mathsf{z}} g_m^2}{2} \right), \tag{4.73}$$

and recall that $v_{mk} = \exp(\mathrm{j} g_m z_{mk})$ from (4.26). Then the mth term of the sum in the objective in (4.71) becomes

$$\begin{aligned} &\int_{\mathbb{R}^K} \mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{z}}_m, \boldsymbol{Q}_m^{\mathsf{z}}) \left| y_m - \sum_{k=1}^{K} \alpha_k q_{mk} v_{mk} \right|^2 \mathrm{d}\boldsymbol{z}_m \\ &= |y_m|^2 - 2 \sum_{k=1}^{K} \alpha_k q_{mk} \operatorname{Re}\left\{ y_m^* \rho_{mk} \right\} \\ &\quad + \sum_{k=1}^{K} \alpha_k q_{mk} \rho_{mk}^* \sum_{l\neq k}^{K} \alpha_l q_{ml} \rho_{ml} + \sum_{k=1}^{K} \alpha_k^2 q_{mk}^2, \end{aligned} \tag{4.74}$$

where we used the fact that $\int_{\mathbb{R}} \mathcal{N}(z_{mk}; \widehat{z}_{mk}, q^{\mathsf{z}}_{mk}) v_{mk}\, \mathrm{d}z_{mk} = \rho_{mk}$. After reapplying the sum over m, we get

$$\frac{\partial}{\partial \alpha_k} \sum_{m=1}^{M} \int_{\mathbb{R}^K} \mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{z}}_m, \boldsymbol{Q}^{\mathsf{z}}_m) \left| y_m - \sum_{k=1}^{K} \alpha_k q_{mk} v_{mk} \right|^2 \mathrm{d}\boldsymbol{z}_m$$
$$= -2 \sum_{m=1}^{M} q_{mk} \gamma_{mk} \tag{4.75}$$
$$\frac{\partial}{\partial \tau_k} \sum_{m=1}^{M} \int_{\mathbb{R}^K} \mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{z}}_m, \boldsymbol{Q}^{\mathsf{z}}_m) \left| y_m - \sum_{k=1}^{K} \alpha_k q_{mk} v_{mk} \right|^2 \mathrm{d}\boldsymbol{z}_m$$
$$= \alpha_k \sum_{m=1}^{M} g_m^2 q_{mk} \gamma_{mk} \tag{4.76}$$

for

$$\gamma_{mk} \triangleq \mathrm{Re}\left\{ y_m^* \rho_{mk} \right\} - \alpha_k q_{mk} - \sum_{l \neq k}^{K} \alpha_l q_{ml} \, \mathrm{Re}\left\{ \rho_{mk}^* \rho_{ml} \right\}. \tag{4.77}$$

We found that complexity of hyperparameter tuning can be substantially reduced, without much loss in accuracy, by using only a subset of the terms in the sum in (4.71), as well as in the corresponding gradient expressions (4.75)-(4.76). For the experiments in Section 4.3, we used a fixed random subset of $\min(M, 20K)$ terms.

4.2.6 Algorithm Summary

Table 4.2 summarizes the CL-AMP algorithm with R random initializations and tuning of the hyperparameters $(\boldsymbol{\alpha}, \boldsymbol{\tau})$. Note that the random initializations $\{\widehat{\boldsymbol{X}}_{0,r}\}$ are used only for the first EM iteration, i.e., $i = 0$. Subsequent EM iterations (i.e., $i \geq 1$) are initialized using the output $\widehat{\boldsymbol{X}}_i$ of the previous EM iteration.

```
Require: Measurements $\boldsymbol{y} \in \mathbb{C}^M$, gains $\{g_m\}_{m=1}^M$, number of initializations $R \geq 1$, initializations
    $\{\widehat{\boldsymbol{X}}_{0,r}\}_{r=1}^R$, $q_0^{\mathsf{p}}$, $\boldsymbol{\alpha}_0$, $\boldsymbol{\tau}_0$.
 1: $i = 0$
 2: repeat
 3:     if $i = 0$ then
 4:         for $r = 1 : R$ do
 5:             Run CL-AMP with fixed $(\boldsymbol{\alpha}_0, \boldsymbol{\tau}_0)$ from initialization $(\widehat{\boldsymbol{X}}_{0,r}, q_0^{\mathsf{p}})$, yielding output $\widehat{\boldsymbol{X}}_{1,r}$,
                $\widehat{\boldsymbol{Z}}_r$, and $\{q_{mr}^{\mathsf{z}}\}_{m=1}^M$.
 6:         end for
 7:         Compute $\widehat{y}_{mr} \triangleq \sum_{k=1}^K \alpha_{0k} \exp(-g_m^2 \tau_{0k}) \exp(\mathrm{j} g_m \widehat{z}_{mkr})\ \forall mr$
 8:         Find $r_* = \arg\min_r \|\boldsymbol{y} - \widehat{\boldsymbol{y}}_r\|$.
 9:         Set $\widehat{\boldsymbol{X}}_1 = \widehat{\boldsymbol{X}}_{1,r_*}$, $\widehat{\boldsymbol{Z}} = \widehat{\boldsymbol{Z}}_{r_*}$ and $\{q_m^{\mathsf{z}}\}_{m=1}^M = \{q_{mr_*}^{\mathsf{z}}\}_{m=1}^M$.
10:     else
11:         Run CL-AMP with fixed $(\boldsymbol{\alpha}_i, \boldsymbol{\tau}_i)$ from initialization $(\widehat{\boldsymbol{X}}_i, q_0^{\mathsf{p}})$, yielding output $\widehat{\boldsymbol{X}}_{i+1}$, $\widehat{\boldsymbol{Z}}$,
            and $\{q_m^{\mathsf{z}}\}_{m=1}^M$.
12:     end if
13:     Compute $(\boldsymbol{\alpha}_{i+1}, \boldsymbol{\tau}_{i+1})$ via (4.71) using $\widehat{\boldsymbol{Z}}$ and $\{q_m^{\mathsf{z}}\}_{m=1}^M$.
14:     $i \leftarrow i + 1$.
15: until convergence
```

Table 4.2: CL-AMP with hyperparameter tuning and multiple random initializations

4.2.7 Frequency Generation

As proposed in [55], $\boldsymbol{a}_m$ were drawn uniformly on the unit sphere and $\{g_m\}$ were drawn i.i.d. from the distribution

$$p(g; \sigma^2) \propto 1_{[0,\infty)}(g) \sqrt{g^2\sigma^2 + \frac{g^4\sigma^4}{4}} \exp\Big(-\frac{1}{2} g^2 \sigma^2\Big), \tag{4.78}$$

which has parameter σ^2. The authors in [55] suggest using $\sigma^2 = \frac{1}{NK}\sum_{k=1}^K \operatorname{tr}(\boldsymbol{\Phi}_k)$ and propose a method to estimate σ^2 from $\boldsymbol{y}$. However, our numerical experiments suggest that using

$$\sigma^2 = \mathrm{E}\{\|\boldsymbol{d}\|_2^2\}/N \approx \|\boldsymbol{D}\|_F^2/NT \tag{4.79}$$

provides significantly improved performance. Note that the right side of (4.79) can be computed in an online manner, or approximated using a subset of the data. For the experiments in Section 4.3, we computed σ^2 via the right side of (4.79) and used it in computing $q_0^{\mathsf{x}} = \sigma^2 \mathbf{1}$.

4.3 Numerical Experiments

In this section, we present the results of several experiments used to test the performance of the CL-AMP, CL-OMPR, and k-means++ algorithms. For k-means++, we used the implementation provided by MATLAB and, for CL-OMPR, we downloaded the MATLAB implementation from [64]. CL-OMPR and CL-AMP used the same sketch $\boldsymbol{y}$, whose frequency vectors $\boldsymbol{W}$ were drawn using the method described in Section 4.2.7, with the scaling parameter σ^2 set via (4.79). For CL-OMPR and CL-AMP, the reported runtimes include the time of computing the sketch, unless otherwise noted. All experiments were run on a Dell PowerEdge C6320 two-socket server with Intel Xeon E5-2680 v4 processors (14 cores, 2.40GHz) and 128GB RAM.

4.3.1 Experiments with Synthetic Data

Performance vs. sketch length M

In the first experiment, we test each algorithm's ability to minimize SSE on a set of training data, i.e., to solve the problem (4.1). In addition, we test how well the recovered centroids work in minimum-distance classification.

The experiment was conducted as follows. Fixing the number of classes at $K = 10$ and the data dimension at $N = 100$, 10 Monte Carlo trials were performed. In each trial, the true centroids were randomly drawn[15] as $\boldsymbol{x}_k \sim \mathcal{N}(\mathbf{0}_N, 1.5^2 K^{2/N} \boldsymbol{I}_N)$. Then, using these centroids, a training dataset $\{\boldsymbol{d}_t\}_{t=1}^{T}$ with $T = 10^7$ samples was drawn from

[15]This data-generation model was chosen to match that from [55], and is intended to have a relatively constant Bayes error rate w.r.t. N and K.

the GMM (4.5) with weights $\alpha_k = 1/K$ and covariances $\boldsymbol{\Phi}_k = \boldsymbol{I}_N \forall k$. Additionally, a test dataset $\{\overline{\boldsymbol{d}}_t\}$ of 10^6 samples was independently generated.

For centroid recovery, k-means++ was invoked on the training dataset, and both CL-AMP and CL-OMPR were invoked after sketching the training data with M samples as in (4.2). Sketch lengths $M/KN \in \{1, 2, 3, 5, 10, 20\}$ were investigated. CL-AMP used two random initializations, i.e., $R = 2$ as defined in Table 4.2.

For each algorithm, the SSE of its estimated centroids $\{\widehat{\boldsymbol{x}}_k\}_{k=1}^K$ was calculated using the training data $\{\boldsymbol{d}_t\}_{t=1}^T$ via (4.1). Additionally, the performance of the estimated centroids in minimum-distance classification was evaluated as follows. First, labels $\{j_k\}_{k=1}^K$ were assigned to the estimated centroids by solving the linear assignment problem [65] without replacement, given by

$$\underset{\{j_1,\dots,j_K\}=\{1,\dots,K\}}{\arg\min} \sum_{k=1}^{K} \|\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_{j_k}\|_2^2. \tag{4.80}$$

Next, each test sample $\overline{\boldsymbol{d}}_t$ was classified using minimum-distance classification, producing the estimated label

$$\widehat{k}_t = \underset{k\in\{1,\dots,K\}}{\arg\min} \|\overline{\boldsymbol{d}}_t - \widehat{\boldsymbol{x}}_{j_k}\|. \tag{4.81}$$

The classification error rate (CER) was then calculated as the proportion of estimated labels $\widehat{k}_t$ that do not equal the true label k_t from which the test sample $\overline{\boldsymbol{d}}_t$ was generated.[16]

Figures 4.1, 4.2, and 4.3 show the median SSE, CER, and runtime (including sketching), respectively, for CL-AMP and CL-OMPR versus M/KN. Also shown is the median SSE, CER, and runtime of k-means++, as a baseline, where k-means++

[16]Note that the true label k_t was assigned when the test sample $\overline{\boldsymbol{d}}_t$ was generated. The true label k_t does not necessarily indicate which of the true centroids $\{\boldsymbol{x}_k\}$ is closest to $\overline{\boldsymbol{d}}_t$.

has no dependence on M. Because a low runtime is meaningless if the corresponding SSE is very high, the runtime was not shown for CL-AMP and CL-OMPR whenever its SSE was more than 1.5 times that of k-means++. The error bars show the standard deviation of the estimates.

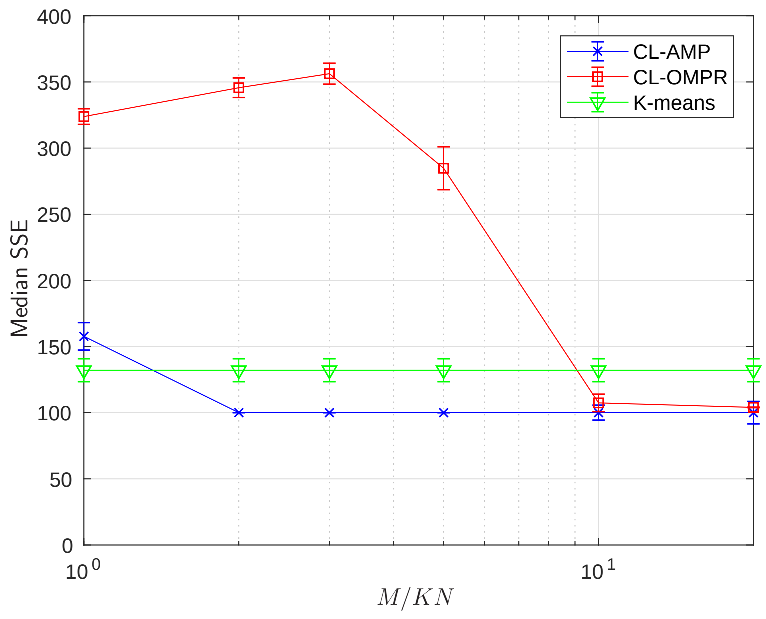


Figure 4.1: SSE vs. sketch length M for $K = 10$ clusters, dimension $N = 100$, and $T = 10^7$ training samples.

Figure 4.1 shows that CL-AMP achieved a low SSE with smaller sketch size M than CL-OMPR. In particular, CL-AMP required $M \approx 2KN$ to yield a low SSE, while CL-OMPR required $M \approx 10KN$. Also, with sufficiently large M, the SSE achieved by CL-AMP and CL-OMPR was lower than that achieved by k-means++.

Figure 4.2 shows that CL-AMP achieved a low CER with sketch size $M \approx KN$, while CL-OMPR required $M \approx 10KN$. Also, with sufficiently large M, CL-AMP

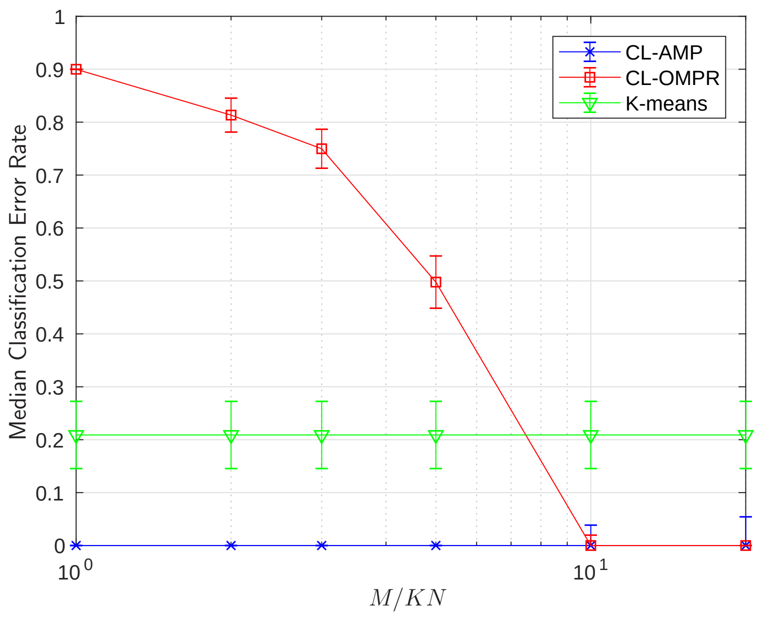


Figure 4.2: Classification error rate vs. sketch length M for $K = 10$ clusters, dimension $N = 100$, and $T = 10^7$ training samples.

and CL-OMPR achieved near-zero CER, whereas k-means++ achieved an error rate of only ≈ 0.2.

Finally, Figure 4.3 shows that, for $M/KN \in \{10, 20\}$, k-means++ ran slightly faster than CL-AMP, which ran slightly faster than CL-OMPR. However, for $M/KN \in \{1, 2, 3, 5\}$, CL-AMP ran significantly faster than k-means++. For $M/KN \in \{1, 2, 3, 5\}$, the runtime of CL-OMPR was not shown because it generated centroids of significantly worse SSE than those of k-means++.

Performance vs. number of classes K

In a second experiment, we evaluated each algorithm's performance versus the number of classes $K \in \{5, 10, 15, 20, 25, 30, 40, 50\}$ and sketch sizes $M/KN \in \{2, 5, 10\}$

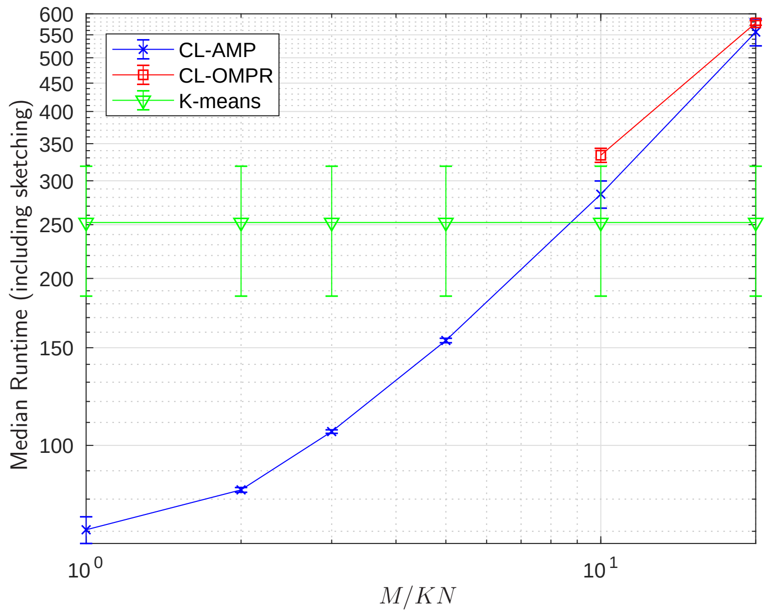


Figure 4.3: Runtime (including sketching) vs. sketch length M for $K = 10$ clusters, dimension $N = 100$, and $T = 10^7$ training samples.

for fixed data dimension $N = 50$. The data was generated in exactly the same way as the previous experiment, and the same performance metrics were evaluated. Figures 4.4, 4.5, and 4.6 show the median SSE, CER, and runtime (including sketching), respectively, versus K, for CL-AMP, CL-OMPR, and k-means++.

Figure 4.4 shows that, as K increases, the SSE of k-means++ remained roughly constant, as expected based on the generation of the true centers $\boldsymbol{x}_k$. For $K \leq 20$, CL-AMP yielded the best SSE for all tested values of M. For $K > 20$, CL-AMP yielded the best SSE with sketch sizes $M \in \{5KN, 10KN\}$, but performed poorly with $M = 2KN$. Meanwhile, CL-OMPR performed reasonably well with sketch size $M = 10KN$, but poorly with $M \in \{2KN, 5KN\}$.

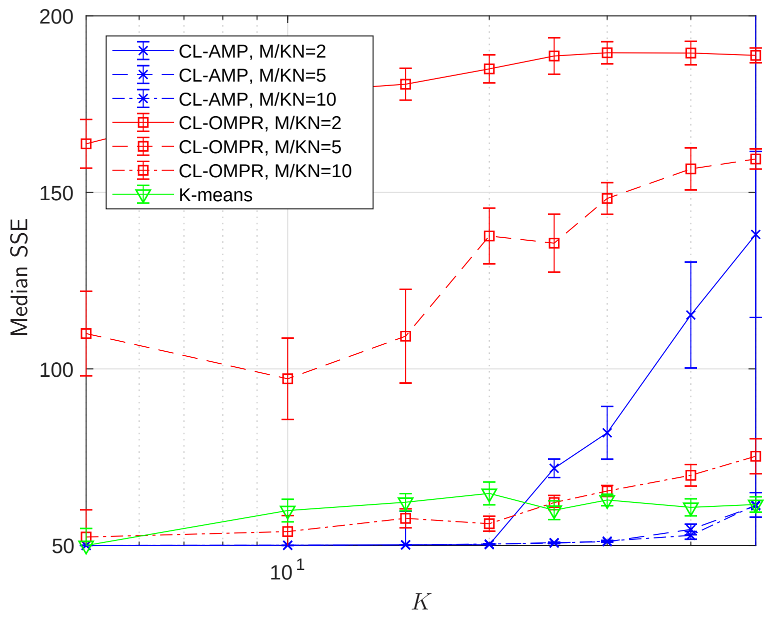


Figure 4.4: SSE vs. number of clusters K for dimension $N = 50$, sketch size $M \in \{2, 5, 10\} \times KN$, and $T = 10^7$ training samples.

Figure 4.5 shows similar trends. With sketch size $M \in \{5KN, 10KN\}$, CL-AMP had the lowest CER of any algorithm for all tested values of K. With sketch size $M = 10KN$, CL-OMPR gave CER better than k-means++ for all tested K, but with $M \in \{2KN, 5KN\}$ CL-OMPR gave CER worse than k-means++ for all tested K.

Finally, Figure 4.6 shows that CL-AMP ran faster than CL-OMPR at all tested K due to its ability to work with a smaller sketch size M. For large K, Figure 4.6 suggests that the runtime of both CL-AMP and CL-OMPR grow as $O(K^2)$. The $O(K^2)$ complexity scaling is expected for CL-AMP, since its complexity is $O(MNK)$ and we set $M = O(K)$. But the $O(K^2)$ complexity scaling is somewhat surprising for CL-OMPR, since its complexity is $O(MNK^2)$ and we set $M = 10NK$. Also, Figure 4.6 shows that CL-AMP ran faster than k-means++ for most values of K; for

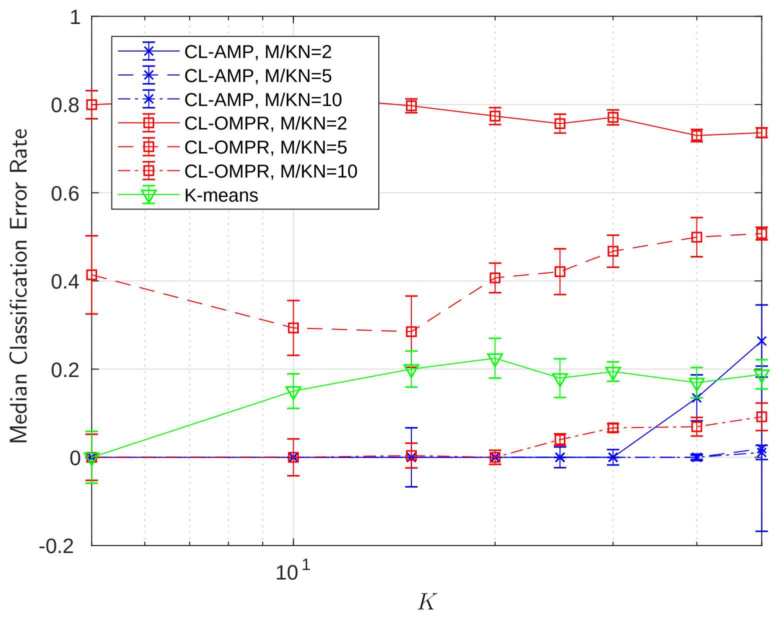


Figure 4.5: Classification Error Rate vs. number of clusters K for dimension $N = 50$, sketch size $M \in \{2, 5, 10\} \times KN$, and $T = 10^7$ training samples.

the smallest tested value of K (i.e., $K = 5$), the median runtime of k-means++ was lower than CL-AMP (but the error-bar suggests that the runtime of k-means++ was highly variable at this K). For the largest tested value of K, k-means++ was again faster than CL-AMP, because the runtime of k-means++ is expected to grow linearly with K, whereas that of CL-AMP is expected to grow quadratically with K when M/KN is fixed.

Performance vs. dimension N

In a third experiment, we evaluated each algorithm's performance versus the dimension N (logarithmically spaced between 10 and 316) for $K = 10$ classes and sketch size $M \in \{2, 5, 10\} \times KN$. The data was generated in exactly the same way as the

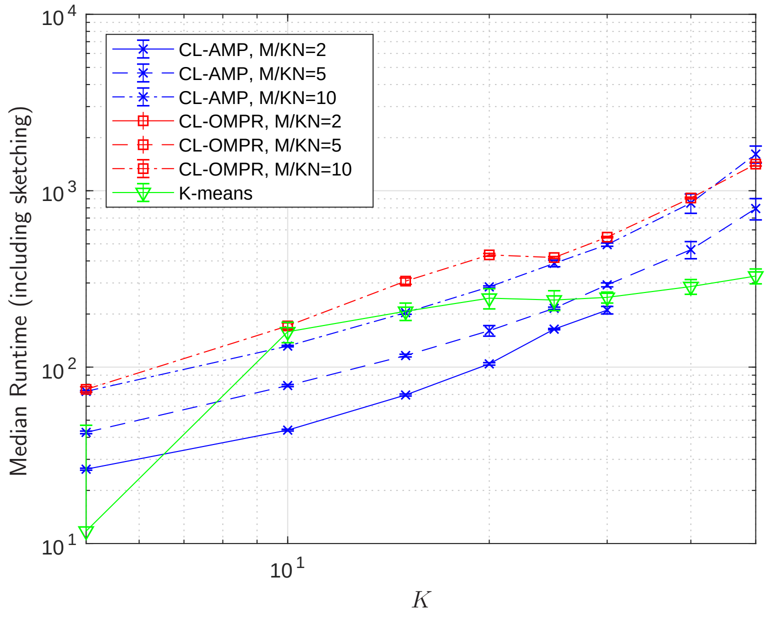


Figure 4.6: Runtime (including sketching) vs. number of clusters K for dimension $N = 50$, sketch size $M \in \{2, 5, 10\} \times KN$, and $T = 10^7$ training samples.

previous two experiments, and the same performance metrics were evaluated. Figures 4.7, 4.8, and 4.9 show the median SSE/N, the CER, and the runtime (including sketching), respectively, versus N, for CL-AMP, CL-OMPR, and k-means++.

Figure 4.7 shows that, among all algorithms, CL-AMP achieved the lowest SSE for all tested values of N and M. Meanwhile, both CL-OMPR under sketch size $M = 10KN$ and k-means++ achieved reasonably good SSE, but CL-OMPR under smaller sketches gave much higher SSE.

Figure 4.8 shows that, among all algorithms, CL-AMP achieved the lowest CER for all tested values of N and M. Meanwhile, CL-OMPR under sketch size $M = 10KN$ gave similar CER to CL-AMP for most N, k-means++ gave significantly worse

CER compared to CL-AMP for all N, and CL-OMPR under sketch size $M = 5KN$ or $2KN$ gave even worse CER for all N.

Finally, Figure 4.9 shows that, among all algorithms, CL-AMP with sketch size $M = 2KN$ ran the fastest for all tested values of N. Meanwhile, CL-OMPR with sketch size $M = 10KN$ ran at a similar speed to CL-AMP with sketch size $M = 10KN$, for all N. The runtimes for CL-OMPR with smaller sketches are not shown because it achieved significantly worse SSE than k-means++. Figure 4.9 suggests that, if N is increased beyond 316, then eventually k-means++ will be faster than CL-AMP under fixed M/KN.

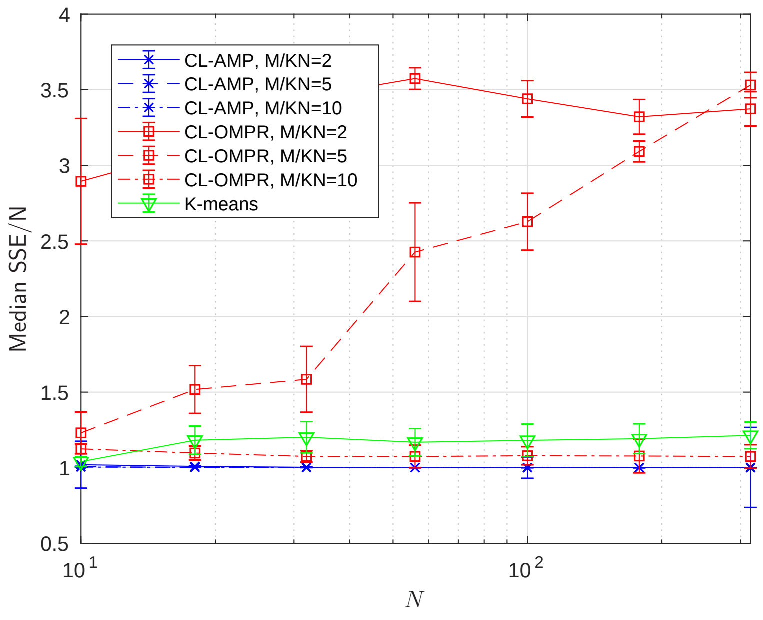


Figure 4.7: SSE/N vs. dimension N for $K = 10$ classes, $T = 10^7$ samples, and sketch size $M \in \{2, 5, 10\} \times KN$.

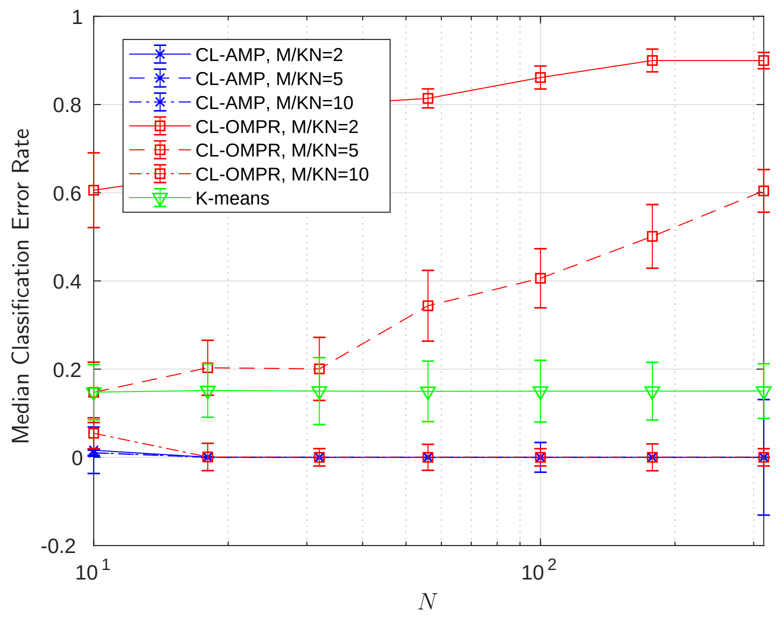


Figure 4.8: Classification Error Rate vs. dimension N for $K = 10$ classes, $T = 10^7$ samples, and sketch size $M \in \{2, 5, 10\} \times KN$.

Performance vs. training size T

In a final synthetic-data experiment, we evaluated each algorithm's performance versus the number of training samples T (logarithmically spaced between 10^5 and 10^8) for $K = 10$ classes, dimension $N = 50$, and sketch size $M \in \{2, 5, 10\}KN$. The data was generated in exactly the same way as the previous three experiments, and the same performance metrics were evaluated.

Figure 4.10 shows the median SSE and CER versus T, for CL-AMP, CL-OMPR, and k-means++. From these figures, we observe that the SSE and CER for each algorithm (and sketch length M) were approximately invariant to T. CL-AMP (under any tested M) yielded the lowest values of SSE and CER. Both CL-OMPR under

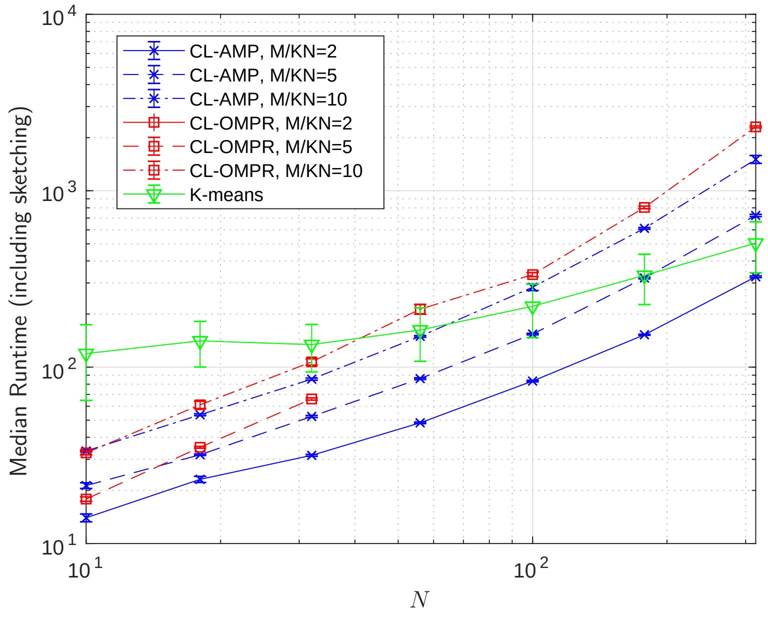
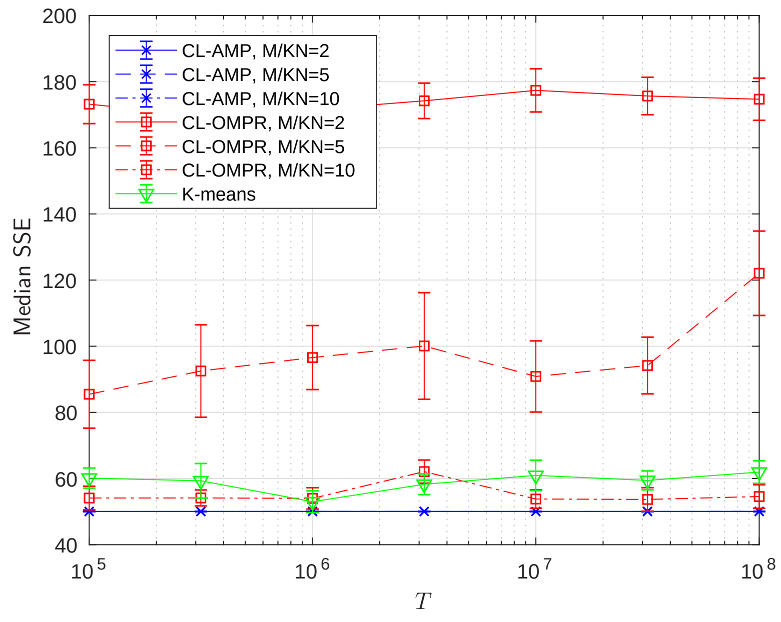


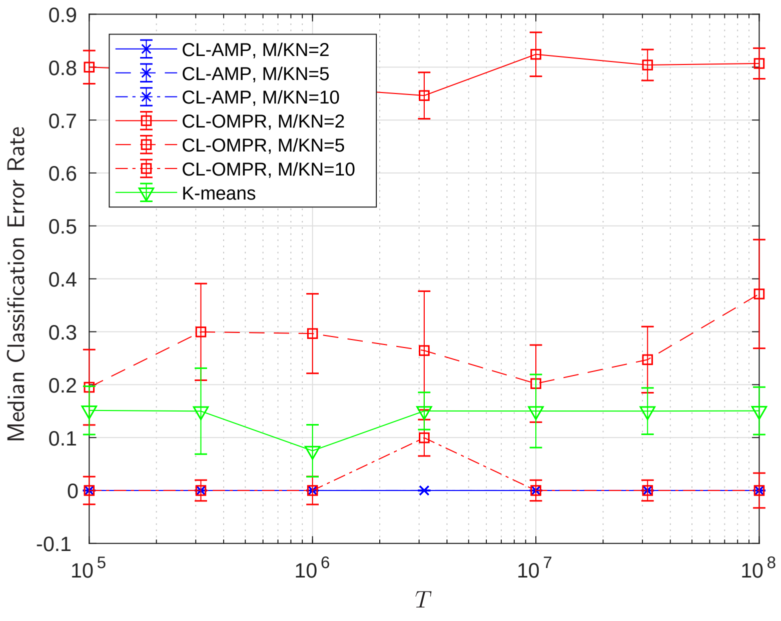
Figure 4.9: Runtime (including sketching) vs. dimension N for $K = 10$ classes, $T = 10^7$ samples, and sketch size $M \in \{2, 5, 10\} \times KN$.

sketch size $M = 10KN$ and k-means++ gave reasonably good SSE and CER, but CL-OMPR under smaller sketches gave worse SSE and CER.

Then, Figure 4.11 shows the median runtime with and without sketching, respectively, for the algorithms under test. Figure 4.11a shows that, if sketching time is included in runtime, then all runtimes increased linearly with training size T. However, for large T, CL-AMP ran faster than k-means++ and CL-OMPR (while also achieving lower SSE and CER). Meanwhile, Figure 4.11b shows that, if sketching time is not included in runtime, then the runtimes of both CL-AMP and CL-OMPR were relatively invariant to T. Also, Figure 4.11 shows that, for $T > 10^6$, the sketching time was the dominant contributer to the overall runtime.

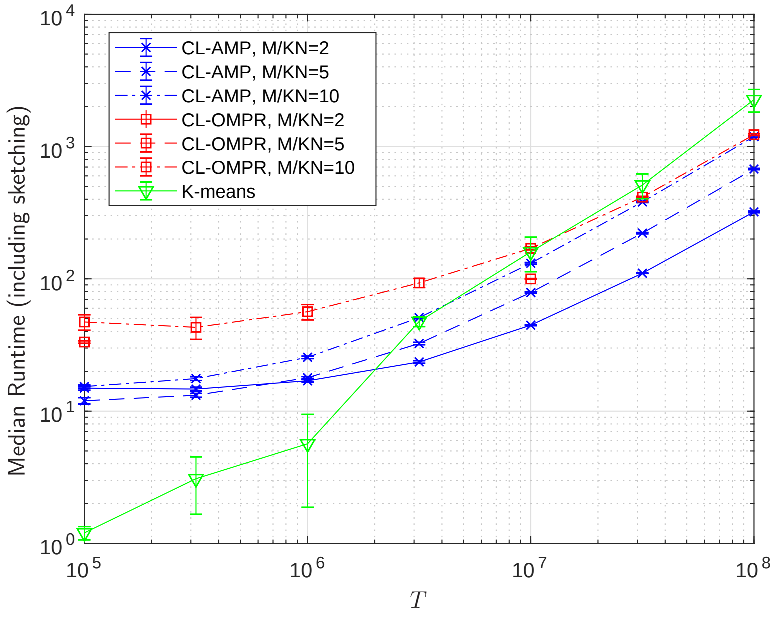


(a) SSE vs. T

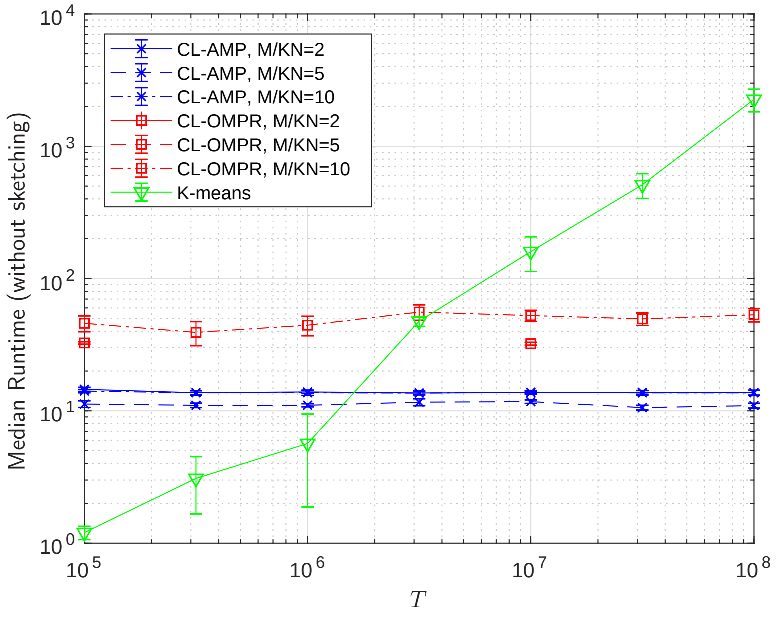


(b) Classification Error Rate vs. T

Figure 4.10: Performance vs. training size T for $K = 10$ classes, dimension $N = 50$, and sketch size $M \in \{2, 5, 10\} \times KN$.



(a) Runtime (including sketching) vs. T



(b) Runtime (without sketching) vs. T

Figure 4.11: Runtime vs. training size T for $K = 10$ classes, dimension $N = 50$, and sketch size $M \in \{2, 5, 10\} \times KN$.

4.3.2 Spectral Clustering of MNIST

Next we evaluated the algorithms on the task of spectral clustering [66] of the MNIST dataset. This task was previously investigated for CL-OMPR and k-means++ in [56], and we used the same data preprocessing steps: extract SIFT descriptors [67] of each image, compute the K-nearest-neighbors adjacency matrix (for $K = 10$) using FLANN [68], and compute the 10 principal eigenvectors of the associated normalized Laplacian matrix (since we know $K = 10$), yielding features of dimension $N = 10$. We applied this process to the original MNIST dataset, which includes $T = 7 \times 10^4$ samples, as well as an augmented one with $T = 3 \times 10^5$ samples constructed as described in [56].

The experiment was conducted as follows. In each of 10 trials, we randomly partitioned each sub-dataset into equally-sized training and testing portions. Then, we invoked CL-AMP, CL-OMPR, and k-means++ on the training portion of the dataset, using sketch sizes $M \in \{1, 2, 3, 5, 10\} \times KN$ for CL-AMP and CL-OMPR. The algorithm parameters were the same as in Section 4.3.1. Finally, the estimated centroids produced by each algorithm were evaluated using the same two metrics as in Section 4.3.1: SSE on the training data, and classification error rate (CER) when the centroids were used for minimum-distance classification of the test data samples.

The median SSE, CER, and runtime, versus sketch length M, are shown for CL-AMP and CL-OMPR in Figures 4.12-4.14, respectively, for the $T = 7 \times 10^4$-sample MNIST sub-dataset, and in Figures 4.15-4.17, respectively, for $T = 3 \times 10^5$-sample MNIST sub-dataset. As before, k-means++ is shown, as a baseline, although it does not use the sketch and thus is performance is invariant to M. From these figures, we observe that CL-AMP and CL-OMPR gave respectable results for sketch lengths $M \geq 2KN$, and SSE nearly identical to kmeans++ for $M \geq 5KN$. For $M \geq 2KN$, however, CL-AMP yielded significantly lower CER than both CL-OMPR and k-means++, at the cost of a slower runtime. We attribute CL-AMP's slower runtime to its use of many iterations i in Table 4.2 for hyperparameter tuning.

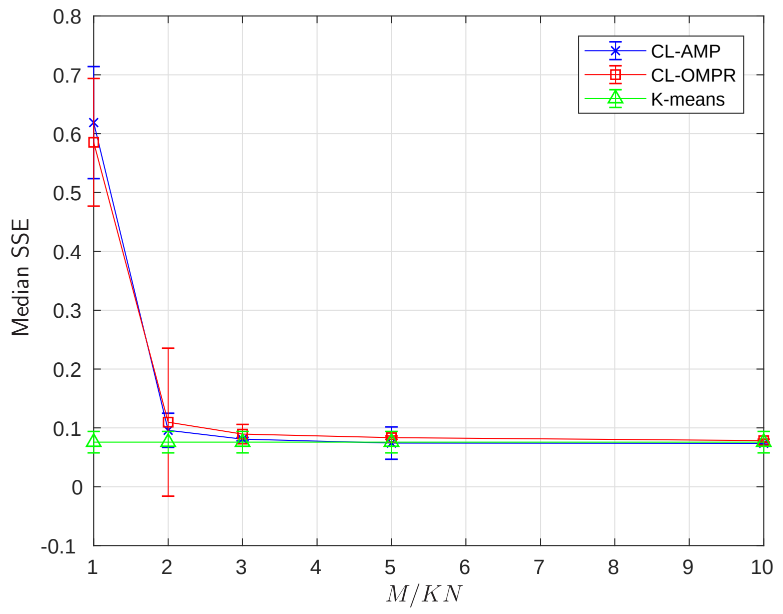


Figure 4.12: SSE vs. M for the $T = 70\,000$-sample spectral MNIST dataset, with $K = 10$ clusters and dimension $N = 10$.

4.3.3 Frequency Estimation

Our final experiment concerns multi-dimensional frequency estimation. Consider a sum-of-sinusoids signal of the form

$$y(\boldsymbol{t}) = \sum_{k=1}^{K} \alpha_k \exp(\mathrm{j}\boldsymbol{t}^\mathsf{T}\boldsymbol{x}_k), \tag{4.82}$$

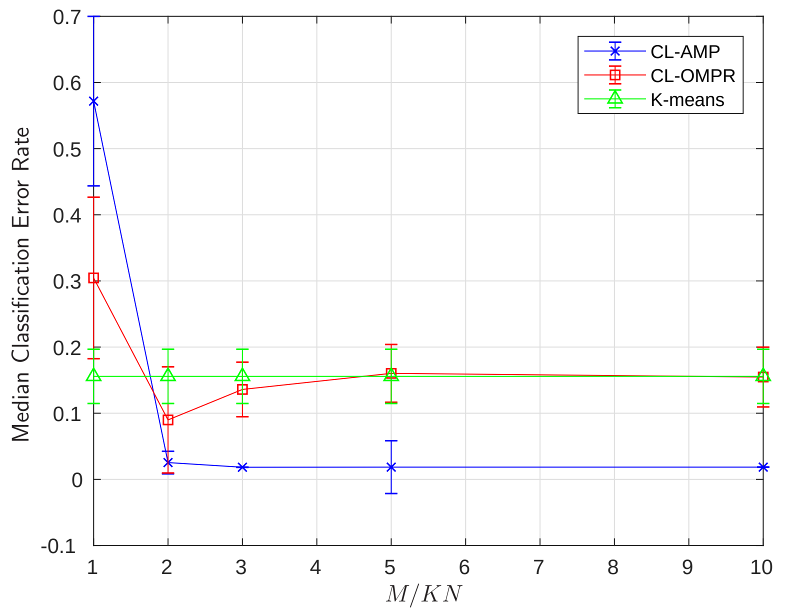


Figure 4.13: Classification Error Rate vs. M for the $T = 70\,000$-sample spectral MNIST dataset, with $K = 10$ clusters and dimension $N = 10$.

where $\boldsymbol{x}_k \in \mathbb{R}^N$ is the frequency of the kth sinusoid, $\alpha_k > 0$ is the amplitude of the kth sinusoid, and $\boldsymbol{t} \in \mathbb{R}^N$ denotes time. Given measurements of the signal $y(\boldsymbol{t})$ at a collection of random times $\boldsymbol{t} \in \{\boldsymbol{t}_m\}_{m=1}^M$, i.e.,

$$y_m = y(\boldsymbol{t}_m) \text{ for } m = 1, \dots, M, \tag{4.83}$$

we seek to recover the frequencies $\{\boldsymbol{x}_k\}_{k=1}^K$. We are particularly interested in the case where the frequencies $\{\boldsymbol{x}_k\}$ are closely spaced, i.e., the "super-resolution" problem.

Note that the model in (4.82) matches that in (4.13) with $g_m \boldsymbol{a}_m = \boldsymbol{t}_m \; \forall m$ and $\boldsymbol{\Phi}_k = \boldsymbol{0} \forall k$, so that we can apply CL-AMP to this frequency estimation problem. The model in (4.82) also matches (4.4) with $\boldsymbol{w}_m = \boldsymbol{t}_m \; \forall m$, and so we can also apply CL-OMPR. But we cannot apply k-means++.

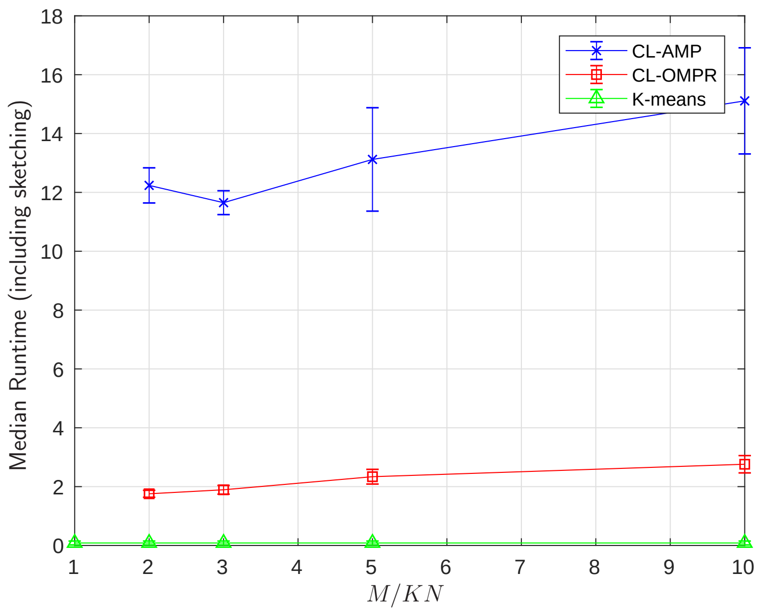


Figure 4.14: Runtime (including sketching) vs. M for the $T = 70\,000$-sample spectral MNIST dataset, with $K = 10$ clusters and dimension $N = 10$.

For frequency pairs $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ with $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 \geq \epsilon$, [69] claims that, with $\{\boldsymbol{w}_m\}$ drawn randomly from an appropriate distribution, one can resolve the frequencies with $M \geq O\Big(\ln(1/\epsilon)\Big)$ measurements. However, choosing $\boldsymbol{w}_m$ uniformly spaced on a grid would require $M \geq O(1/\epsilon)$ measurements. Thus, for a final experiment, similar to those performed in [69], we did the following. For a particular N and K (where K is even for simplicity), we generated $K/2$ pairs of frequencies $\{\boldsymbol{x}_{2k-1}, \boldsymbol{x}_{2k}\}$, where $\|\boldsymbol{x}_{2k-1} - \boldsymbol{x}_{2k}\|_2 = \epsilon$ for $k = 1, ..., K/2$. Then, for a particular realization of $\{\boldsymbol{x}_k\}_{k=1}^K$ and $\{\boldsymbol{w}_m\}_{m=1}^M$, CL-AMP and CL-OMPR were invoked to estimate $\{\widehat{\boldsymbol{x}}_k\}_{k=1}^K$. Recovery was declared successful if

$$\max_k \|\boldsymbol{x}_{j_k} - \widehat{\boldsymbol{x}}_k\|_2 < \epsilon/2, \tag{4.84}$$

where $\{j_k\}_{k=1}^K$ solves the linear assignment problem (4.80).

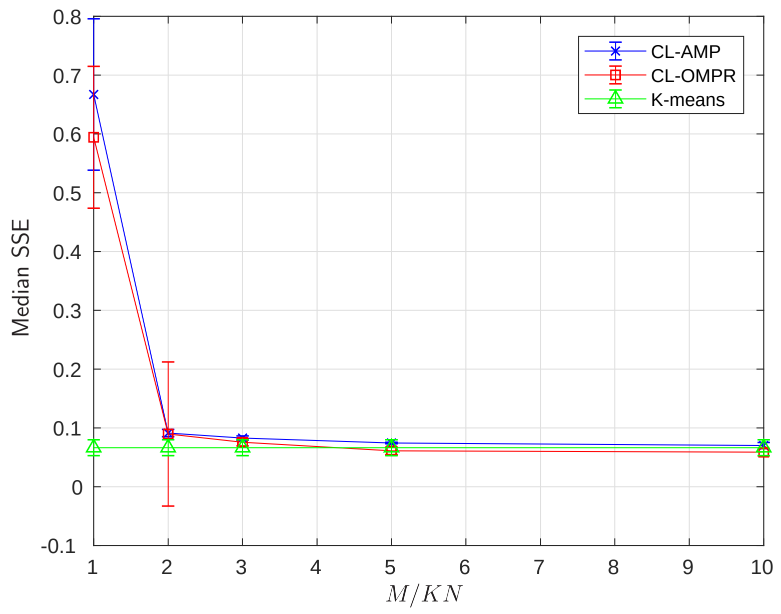


Figure 4.15: SSE vs. M for the $T = 300\,000$-sample spectral MNIST dataset, with $K = 10$ clusters and dimension $N = 10$.

For our experiment, we tested $K=4$ frequency components of dimension $N=2$ and varied M from $3KN$ to $100KN$ while also varying ϵ from 10^{-1} to 10^{-3}. For each combination, 10 trials were performed. The empirical probability of successful recovery is shown in Figures 4.18-4.19. In Figure 4.18, $\boldsymbol{a}_m$ were drawn uniformly on the unit sphere and $g_m = |g'_m|$ with $g'_m \sim \mathcal{N}\Big(0, 4\epsilon^2 \log_{10}^2(\epsilon)\Big)$. Superimposed on the figures are curves showing $M/KN = 0.1/\epsilon$ and $M/KN = \ln(1/\epsilon)$. From the figures, we see that CL-AMP had a higher empirical probability of recovery than CL-OMPR, especially for small ϵ. We also see that the empirical phase transition of CL-AMP is close to the $\ln(1/\epsilon)$ curve with random frequency samples and the $0.1/\epsilon$ curve with uniform frequency samples.

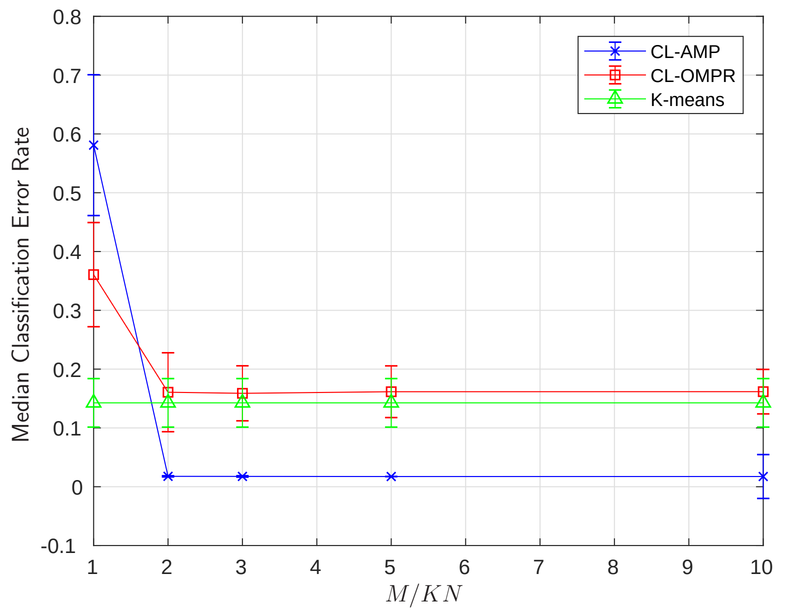


Figure 4.16: Classification Error Rate vs. M for the $T = 300\,000$-sample spectral MNIST dataset, with $K = 10$ clusters and dimension $N = 10$.

4.4 Conclusion

In sketched clustering, the original dataset is sketched down to a relatively short vector, from which the centroids are extracted. For the sketch proposed by [55, 56], we proposed the "CL-AMP" centroid-extraction method. Our method assumes that the original data follows a GMM, and exploits the recently proposed simplified hybrid generalized approximate message passing (SHyGAMP) algorithm [9]. Numerical experiments suggest that CL-AMP exhibits better sample complexity (i.e., extracts accurate clusters with fewer compressed samples) than the state-of-the-art sketched-clustering algorithm, CL-OMPR, from [55,56]. In many cases, CL-AMP also exhibits better computational complexity than CL-OMPR. Furthermore, for datasets with

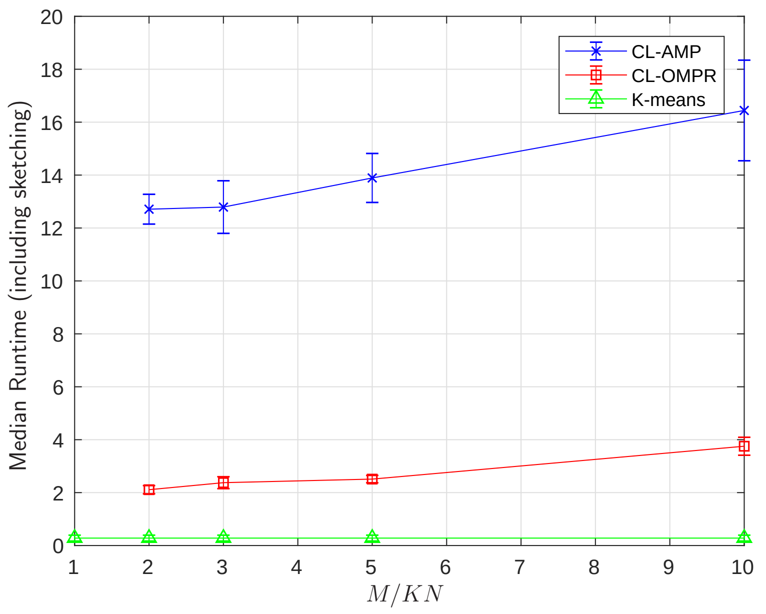
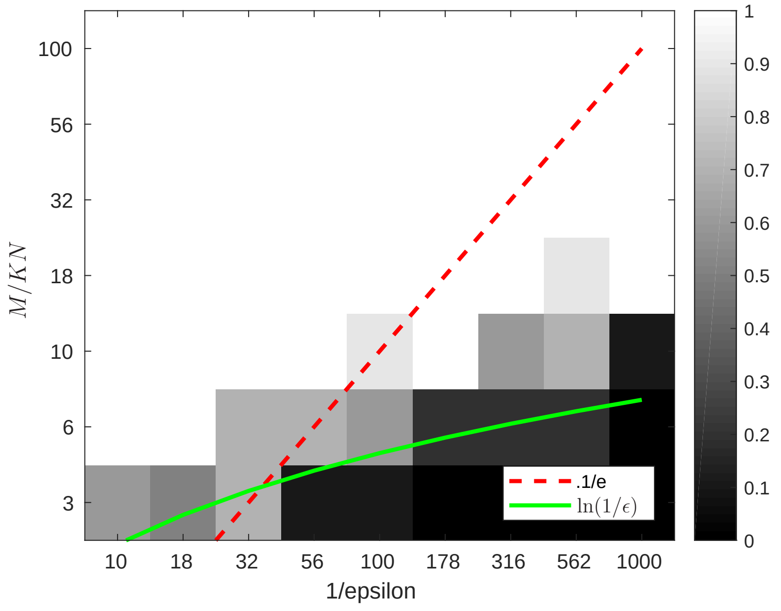
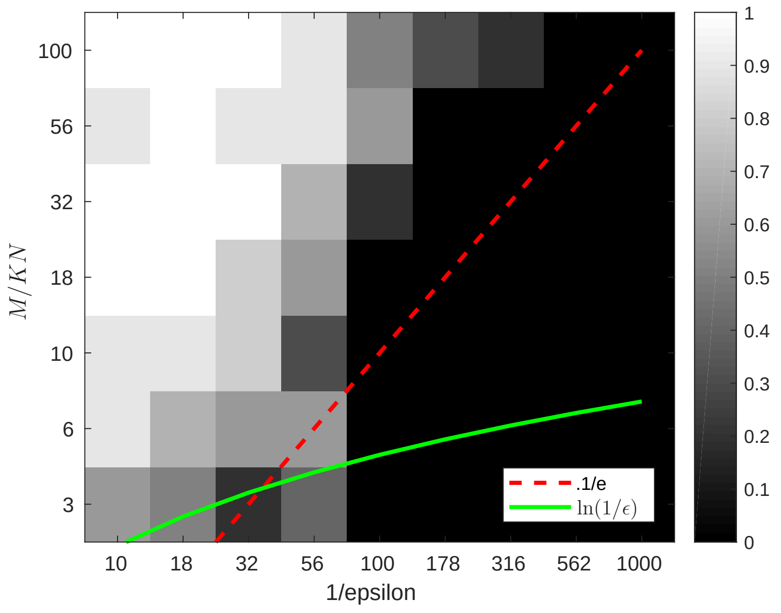


Figure 4.17: Runtime (including sketching) vs. M for the $T = 300\,000$-sample spectral MNIST dataset, with $K = 10$ clusters and dimension $N = 10$.

many samples, CL-AMP exhibits lower computational complexity than the widely used k-means++ algorithm. As future work, it would be interesting to consider the use of fast deterministic sketching with CL-AMP.

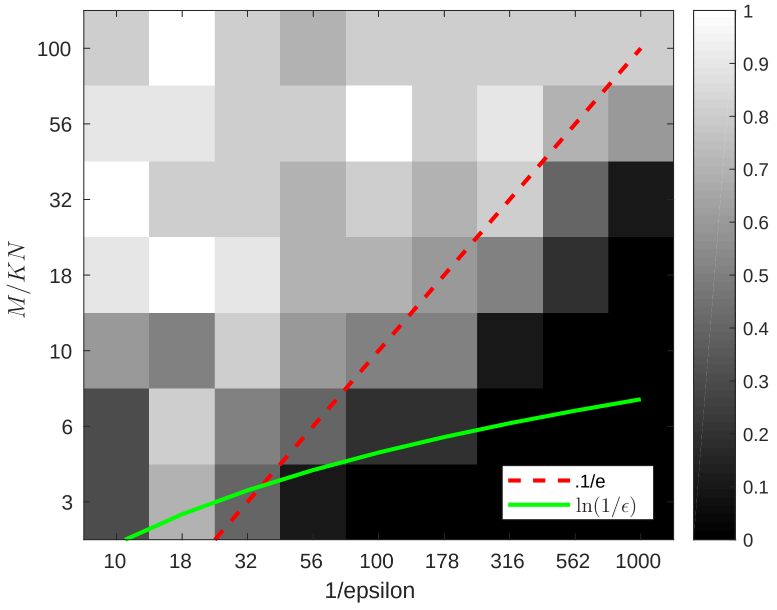


(a) CL-AMP

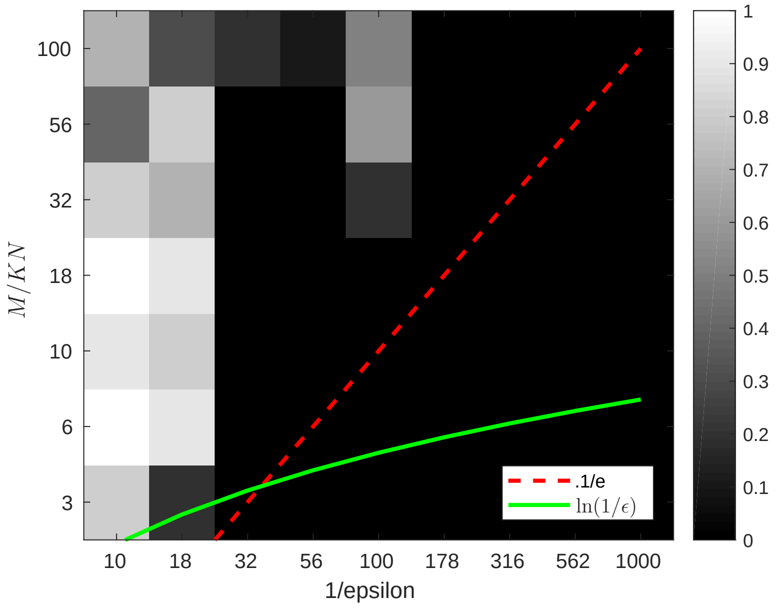


(b) CL-OMPR

Figure 4.18: Frequency estimation for $K = 4$ and $N = 2$ with random time samples.



(a) CL-AMP



(b) CL-OMPR

Figure 4.19: Frequency estimation for $K = 4$ and $N = 2$ with uniformly spaced time samples.

Chapter 5: Adaptive Detection from Quantized Measurements

5.1 Introduction and Motivation

In this final chapter, we consider the problem of adaptive detection from highly quantized measurements, specifically in the case of strong, but low-rank interference. The motivation for studying this problem is the following. In military communications the receiver may experience strong jamming from a few number of sources and in order to effectively null the interference, a large number of antennas at the receiver are required. However, using more antennas increases the hardware complexity due to the increased number of analog to digital converters (ADCs) and related components, as well as increasing the total amount of data that must be processed. For this reason, it may be desirable to use few-bit or even 1-bit ADCs, which may decrease the amount of data to be processed, as well as simplify hardware design at the antenna (1-bit ADCs in particular greatly decrease the hardware complexity at the receiver).

However, using low-resolution ADCs introduces new challenges. In particular, quantization introduces error in the measurements, and it may affect how one chooses to process the data. Existing algorithms for signal detection assume unquantized measurements, and it is unclear how they will perform when applied to quantized data.

In this chapter, we first apply unquantized detection algorithms to quantized data to study how well this approach works. We observe that the primary regime of failure is when the interference power is too large. We then investigate various alternatives to how we process the data in order to have improved detection performance at large interference levels. The various approaches we consider are:

1. standard methods for improving the performance of quantized systems, such as dithering and companding,

2. alternate detection statistics computed using the generalized likelihood ratio test (GLRT) and the true quantization model, and

3. various analog pre-processing techniques to remove interference prior to quantization.

The primary connection between this chapter and the rest of this dissertation is in item 2 above. In particular, we see that the GLRT that uses the true quantization model involves generalized bilinear inference (see Section 1.1.3), for which approximate message passing and other techniques exist.

5.1.1 Problem Statement

Our objective is to determine the presence or absence of a known signal $\boldsymbol{s}$, where the system model under the $\mathcal{H}_1$ and $\mathcal{H}_0$ hypotheses is

$$\mathcal{H}_1 : \boldsymbol{Y} = \mathcal{Q}_{B,\Delta}\Big(\boldsymbol{h}\boldsymbol{s}^{\mathsf{H}} + \boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{H}} + \boldsymbol{W}\Big) \tag{5.1a}$$

$$\mathcal{H}_0 : \boldsymbol{Y} = \mathcal{Q}_{B,\Delta}\Big(\boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{H}} + \boldsymbol{W}\Big), \tag{5.1b}$$

where $\boldsymbol{Y} \in \mathbb{C}^{n_r \times n_s}$ are the quantized measurements, $\boldsymbol{h} \in \mathbb{C}^{n_r}$ is the unknown array response of the signal, $\boldsymbol{s} \in \mathbb{C}^{n_s}$ is the signal, $\boldsymbol{G} \in \mathbb{C}^{n_r \times n_i}$ is the unknown array response of the interference, $\boldsymbol{\Psi} \in \mathbb{C}^{n_s \times n_i}$ is the unknown interference, and $\boldsymbol{W}$ is circular AWGN with variance σ_w^2. The low-rank interference is intended to model a few number of interference sources whose array responses are time invariant. We assume $n_s \geq n_r$, in which case our model can also generalize to the case of full rank interference by assuming $\boldsymbol{G}$ and $\boldsymbol{\Psi}$ both have rank n_r.

For $x \in \mathbb{C}$, $\mathcal{Q}_{B,\Delta}(x) \triangleq \mathcal{Q}_{B,\Delta}(\mathrm{Re}(x)) + \mathrm{j}\mathcal{Q}_{B,\Delta}(\mathrm{Im}(x))$, where

$$\mathcal{Q}_{B,\Delta}(x \in \mathbb{R}) \triangleq \begin{cases} \max\left\{ \min\left\{ \Delta\left(\left\lceil \frac{x}{\Delta} \right\rceil - 1/2 \right), \Delta_c \right\}, -\Delta_c \right\} & \text{if } B \geq 2 \\ \mathrm{sgn}(x) & \text{if } B = 1 \end{cases}, \tag{5.2}$$

where $\Delta_c = \Delta(2^{B-1} - 1/2)$. For $B \geq 2$, (5.2) is the element-wise, B-bit, mid-rise, uniform quantization function with bin size Δ.

5.1.2 Unquantized Detectors

The GLRT for the unquantized model, given by

$$\mathcal{H}_1 : \boldsymbol{U} = \boldsymbol{h}\boldsymbol{s}^{\mathsf{H}} + \boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{H}} + \boldsymbol{W} \tag{5.3a}$$

$$\mathcal{H}_0 : \boldsymbol{U} = \boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{H}} + \boldsymbol{W}, \tag{5.3b}$$

has been studied in depth for various assumptions on the various parameters.

Kelly [70, 71] treated $\boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{H}} + \boldsymbol{W}$ as temporally white and Gaussian with unknown spatial covariance matrix $\boldsymbol{\Sigma} > 0$. In this case, the GLRT is

$$\frac{\max_{\boldsymbol{h},\boldsymbol{\Sigma}} p(\boldsymbol{U}|\mathcal{H}_1; \boldsymbol{h}, \boldsymbol{\Sigma})}{\max_{\boldsymbol{\Sigma}} p(\boldsymbol{U}|\mathcal{H}_0; \boldsymbol{\Sigma})} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau, \tag{5.4}$$

which reduces to

$$\frac{\prod_{n=1}^{n_r} \lambda_{0,n}}{\prod_{n=1}^{n_r} \lambda_{1,n}} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau, \tag{5.5}$$

where $\lambda_{0,n} \geq \lambda_{0,n+1}$ are the decreasing eigenvalues of $\frac{1}{n_s}\boldsymbol{U}\boldsymbol{U}^{\mathsf{H}}$, while $\lambda_{1,n} \geq \lambda_{1,n+1}$ are the decreasing eigenvalues of $\frac{1}{n_s}\boldsymbol{U}\boldsymbol{P}_{\boldsymbol{s}}^{\perp}\boldsymbol{U}^{\mathsf{H}}$, where $\boldsymbol{P}_{\boldsymbol{s}}^{\perp} = \boldsymbol{I}_{n_s} - \boldsymbol{s}(\boldsymbol{s}^{\mathsf{T}}\boldsymbol{s})^{-1}\boldsymbol{s}^{\mathsf{H}}$.

Gerlach and Steiner [72] assumed known σ_w^2 and temporally white Gaussian interference, so that columns of $\boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{H}} + \boldsymbol{W} \sim \mathcal{CN}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\mathsf{LR}} + \sigma_w^2 \boldsymbol{I})$ for unknown low-rank $\boldsymbol{\Sigma}_{\mathsf{LR}} > 0$. The GLRT then includes the constrained maximization

$$\frac{\max_{\boldsymbol{h},\boldsymbol{\Sigma}\in\mathcal{S}_{\sigma_w^2}} p(\boldsymbol{U}|\mathcal{H}_1;\boldsymbol{h},\boldsymbol{\Sigma})}{\max_{\boldsymbol{\Sigma}\in\mathcal{S}_{\sigma_w^2}} p(\boldsymbol{U}|\mathcal{H}_0;\boldsymbol{\Sigma})} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau, \tag{5.6}$$

where $\mathcal{S}_{\sigma_w^2} = \{\boldsymbol{\Sigma}_{\mathsf{LR}} + \sigma_w^2\boldsymbol{I} : \boldsymbol{\Sigma}_{\mathsf{LR}} \geq 0\}$. In this case, the GLRT reduces to the form in (5.5), except with thresholded eigenvalues $\widetilde{\lambda}_{h,n} = \max\{\lambda_{h,n}, \sigma_w^2\}$, for $h \in \{0,1\}$.

Then, Kang, Monga and Rangaswamy (KMR) [73] proposed a variation of [72] where the noise variance σ_w^2 is unknown but the interference rank n_i is known. Their GLRT had the form in (5.6), except now $\mathcal{S}_{\sigma_w^2}$ is replaced with $\mathcal{S}_{n_i} = \{\boldsymbol{\Sigma}_{\mathsf{LR}} + \sigma_w^2\boldsymbol{I} : \operatorname{rank}(\boldsymbol{\Sigma}_{\mathsf{LR}}) = n_i, \boldsymbol{\Sigma}_{\mathsf{LR}} \geq 0, \sigma_w^2 > 0\}$. In this case, the GLRT simplifies to

$$\frac{\prod_{n=1}^{n_r} \widehat{\lambda}_{0,n}}{\prod_{n=1}^{n_r} \widehat{\lambda}_{1,n}} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau, \tag{5.7}$$

where $\{\widehat{\lambda}_{h,n}\}_{n=1}^{n_r}$ are computed via $\widehat{\lambda}_{h,n} = \lambda_{h,n}$ for $n = 1, ..., n_i$ and $\widehat{\lambda}_{h,n} = \widehat{\sigma_{w,h}^2}$ otherwise, where $\widehat{\sigma_{w,h}^2} = \frac{1}{n_r - n_i}\sum_{n=n_i+1}^{n_r} \lambda_{h,n}$ for $h \in \{0,1\}$.

The previous approaches all model the interference as temporally white Gaussian. McWhorter [74] instead proposed to treat $\boldsymbol{G}$ and $\boldsymbol{\Psi}$ as deterministic unknowns, each with n_i columns, and he considered unknown noise variance σ_w^2. In this case, the GLRT simplifies to

$$\frac{\sum_{n=n_i+1}^{n_r} \lambda_{0,n}}{\sum_{n=n_i+1}^{n_r} \lambda_{1,n}} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau. \tag{5.8}$$

5.2 Numerical Study of Unquantized Detectors with Quantized Measurements

Here, we evaluate the detection performance of the Kelly, KMR, and McWhorter detectors when using quantized measurements given by (5.1). We don't include the

Gerlach-Steiner detector in our tests because it assumes known σ_w^2, which we never assume.

In the following simulations, the data are generated iid from the following distributions:

$$\boldsymbol{h} \sim \mathcal{CN}(\mathbf{0}, 1/n_r \boldsymbol{I}_{n_r}), \tag{5.9}$$

$$\text{vec}(\boldsymbol{G}) \sim \mathcal{CN}(\mathbf{0}, 1/n_r \boldsymbol{I}_{n_r n_i}), \tag{5.10}$$

$$\boldsymbol{s} \sim \mathcal{CN}(\mathbf{0}, \boldsymbol{I}_{n_s}) \tag{5.11}$$

$$\text{vec}(\boldsymbol{\Psi}) \sim \mathcal{CN}(\mathbf{0}, \sigma_i^2 \boldsymbol{I}_{n_s n_i}) \tag{5.12}$$

$$\text{vec}(\boldsymbol{W}) \sim \mathcal{CN}(\mathbf{0}, \sigma_w^2 \boldsymbol{I}_{n_r n_s}). \tag{5.13}$$

The following definitions for signal-to-noise-ratio (SNR) and interference-to-signal-ratio (ISR) are also used (noting the signal has unit variance):

$$\text{SNR} \triangleq -10 \log_{10} \left(\sigma_w^2 \right) \text{ dB} \tag{5.14}$$

$$\text{ISR} \triangleq 10 \log_{10} \left(\sigma_i^2 \right) \text{ dB}. \tag{5.15}$$

In the subsequent experiments, unless otherwise stated, for a particular combination of $\{B, \sigma_w^2, \sigma_i^2, n_r, n_s, n_i\}$ we choose $\Delta \in \{\Delta_1, ..., \Delta_n\}$ to maximize the Probability of Detection (P_D) for a given Probability of False Alarm (P_{FA}), where the P_D and P_{FA} are estimated empirically. For a given combination of $\{B, \sigma_w^2, \sigma_i^2, n_r, n_s, n_i\}$, the set $\{\Delta_1, ..., \Delta_n\}$ is chosen in the following manner. First, on a single realization of data generated under the $\mathcal{H}_0$ hypothesis, we find $\Delta_b = \arg\min_\Delta \|\boldsymbol{Y} - \mathcal{Q}_{B,\Delta}(\boldsymbol{Y})\|_F^2$ via a grid search. Then, we generate $\{\Delta_1, ..., \Delta_n\} = \{10^{-2}\Delta_b, 10^{-1.5}\Delta_b, ..., 10\Delta_b\}$. Note that if $B = 1$, the quantization function does not depend on Δ and so this procedure is not required. Also note that Δ_b does not depend on the algorithm under test (e.g., Kelly), but the $\Delta \in \{\Delta_1, ..., \Delta_n\}$ that maximizes P_D is chosen separately for each algorithm under test. Finally, if $B = \infty$, it simply means the measurements were unquantized.

We first test the effect of σ_w^2 and B on detection performance when there is no interference. In Figure 5.1, we plot the estimated P_D of the Kelly detector vs σ_w^2/n_s, when $P_{FA} = 10^{-2}$, and $n_r = 32$, $n_s = 200$, and $\sigma_i^2 = 0$. We show four traces corresponding to $B \in \{1, 2, 3, \infty\}$. We did not include the KMR or McWhorter detectors in this test since there was no interference. Overall, the gain in performance as B increases shrinks to zero, where $B = 3$ and $B = \infty$ have nearly the identical performance.

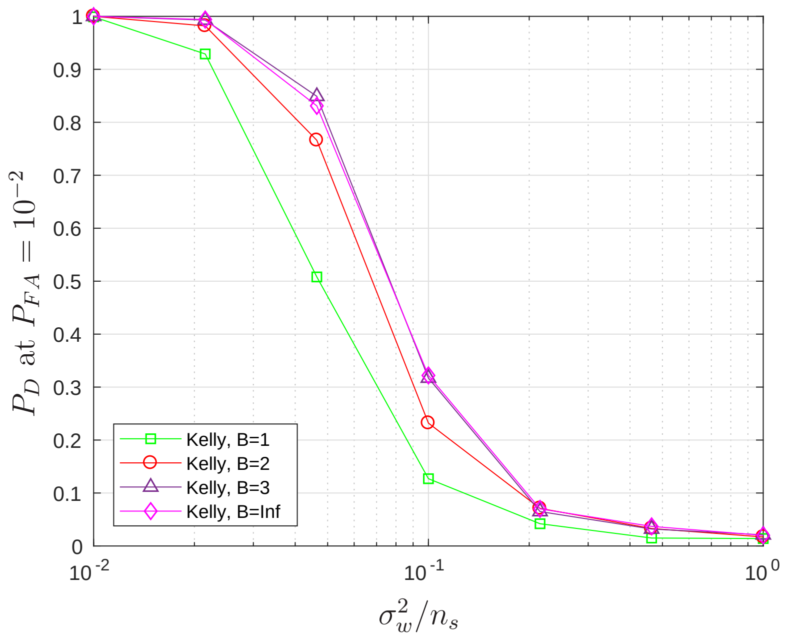


Figure 5.1: Empirical detection probability of the Kelly GLRT vs σ_w^2/n_s in the case of no interference, where $n_r = 16$, $n_s = 100$, $B \in \{1, 2, 3, \infty\}$ and the false alarm probability was 10^{-2}.

An alternate way of looking at this data is given in Figure 5.2. Here, for the same simulation parameters used in Figure 5.1, we plot the horizontal gap between traces of P_D vs σ_w^2/n_s (as in Figure 5.1), relative to the 1-bit trace, for different values of P_D. We refer to this gap as "SNR gain." Calculating this gap requires linearly interpolating the traces to obtain the corresponding σ_w^2 values at the specified P_D values (a less quantized version of the data in Figure 5.1 was used for this purpose with more values of B and σ_w^2 under test, but Figure 5.1 is sufficient for seeing the big picture). Based on this plot, increasing B from 1 to 2 has a similar effect on detection performance as increasing the SNR by 1.5 dB in the 1-bit case. As B increases further, the marginal gains decrease to zero, and eventually the difference in detection performance between a detector with 1-bit and unquantized measurements is approximately equivalent to 2 dB difference in SNR.

Then, we test the effect of the interference power σ_i^2 and bits B on detection performance in the high SNR (low σ_w^2) case. In Figure 5.3 we plot the estimated P_D vs σ_i^2/n_s of the Kelly, KMR, and McWhorter detectors when the $P_{FA} = 10^{-2}$, $n_r = 16$, $n_s = 100$, $n_i = 1$, $\sigma_w^2/n_s = 10^{-2}$ and $B \in \{1, 4, 8, \infty\}$. In this case, the KMR and McWhorter detectors were provided the true value of n_i. One can see that, in the case where $B = \infty$, none of the detection algorithms fail in the range of σ_i^2 values tested, but the detectors where $B < \infty$ do fail, with smaller B failing at a lower σ_i^2. Also, the low-rank KMR and McWhorter detectors perform worse than the Kelly detector. Our explanation for this behavior is that the quantization error is not white, which is the fundamental assumption about the noise for these detectors.

Next, similar to Figure 5.2, in Figure 5.4 we plot the horizontal gap between traces of P_D vs σ_i^2/n_s for the Kelly detector, relative to the 1-bit trace, for different values

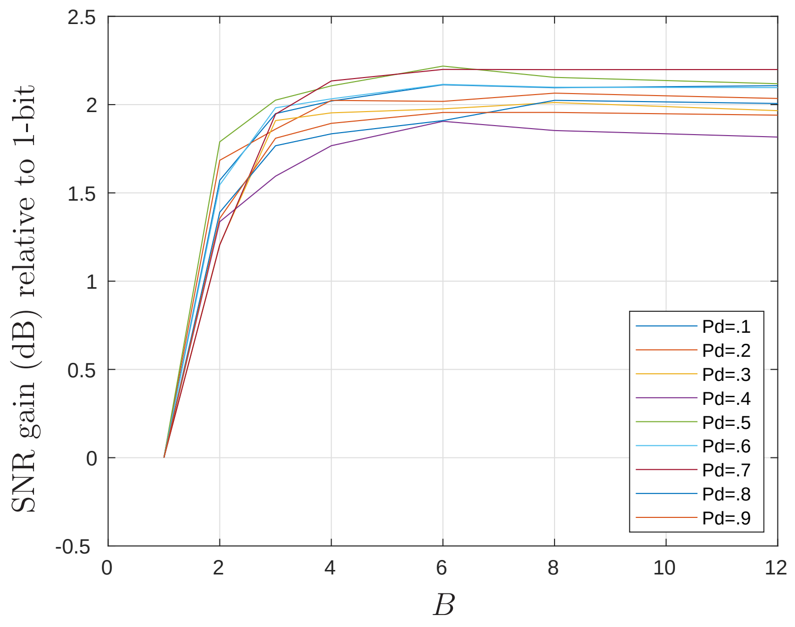


Figure 5.2: SNR gain vs B. Each trace corresponds to a $P_D \in \{.1, .2, ..., .9\}$, where $P_{FA} = 10^{-2}$.

of P_D. We refer to this gap as the "ISR gain." Based on this figure, increasing B by 1 means we can successful handle approximately 5dB more ISR (except the increase from $B = 1$ to 2 has the largest marginal gain). We note that this simulation does not completely isolate the effect of the interference because noise is present and as we saw in Figure 5.2, increasing B has an effect similar to increasing the SNR, which may partly account for the larger increase as B goes from 1 to 2.

5.2.1 Summary

In this section, we observed that approaches to unquantized detection that are given quantized measurements perform adequately well when there is little to no

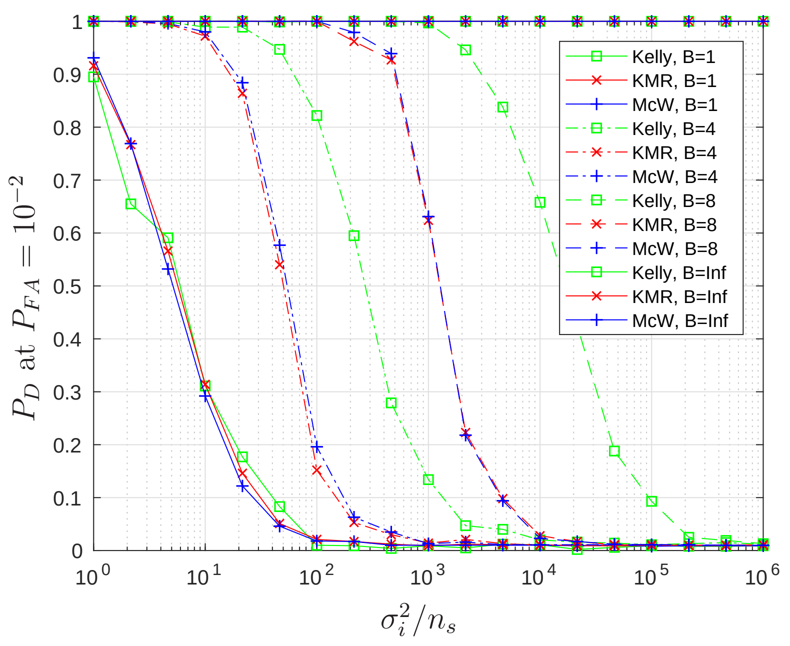


Figure 5.3: Empirical detection probability of the Kelly, KMR, and McWhorter ("McW") GLRTs vs σ_w^2/n_s, where $n_r = 16$, $n_s = 100$, $B \in \{1, 4, 8, \infty\}$, $\sigma_w^2/n_s = 10^{-2}$, and the false alarm probability was 10^{-2}.

interference; however, these methods fail when the interference grows large. In particular, in the interference-free case, the marginal improvement in detection from increasing B eventually goes to zero. However, the amount of interference power that can be handled at a particular (P_D, P_{FA}) is approximately proportional to B.

5.3 Detection Performance with Dither and Companding

In this section we test the detection performance of the Kelly detector applied to quantized measurements (we focus on just the Kelly detector for simplicity) when two conventional techniques to mitigate quantization are applied: dithered quantization and companding.

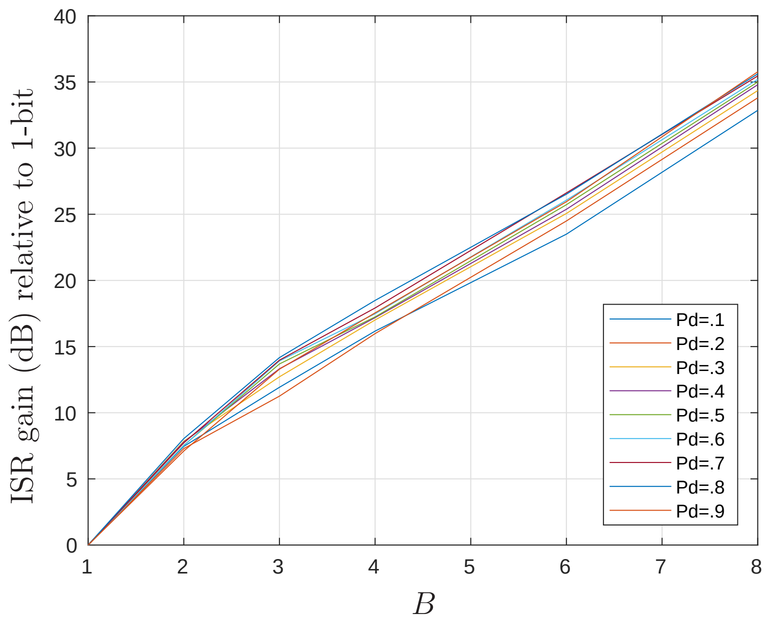


Figure 5.4: ISR gain vs B. Each trace corresponds to a $P_D \in \{.1, .2, ..., .9\}$. where $P_{FA} = 10^{-2}$.

5.3.1 Detection Performance with a Dithered Quantizer

Dither is noise added to the measured signal prior to quantization with the purpose of altering the distribution of the quantization error. In particular, if the dither signal's distribution obeys certain properties, and if the quantizer does not overload, then the quantization error, defined as the difference between the quantizer input and output, is white and uncorrelated with the input [75]. Two commonly used distributions for the dither signal that meet the required properties are iid uniform dither, where $d_n \sim \mathcal{U}(-\Delta/2, \Delta/2)$, and iid triangular dither, where $d_n = d_{n,1} + d_{n,2}$, with $d_{n,i} \sim \mathcal{U}(-\Delta/2, \Delta/2)$ and independent $d_{n,1}$ and $d_{n,2}$.

In Figure 5.5 we plot the P_D of the Kelly detector vs σ_i^2/n_s for various dither distributions when Δ is set so that overload within the quantizer occurred approximately .1% of the time. In this experiment $B \in \{2, 4, 6\}$, $n_r = 16$, $n_s = 100$, $n_i = 1$, $\sigma_w^2/n_s = 10^{-2}$, and $P_{FA} = 10^{-2}$. In our case with complex measurements, we generated independent dither signals for the real and imaginary channels. From this figure we observe that the detection performance with dithering is worse than the detection performance without dithering. Our hypothesized reason for this is that although dither whitens the quantization error, it still increases the overall noise, which is detrimental for detection.

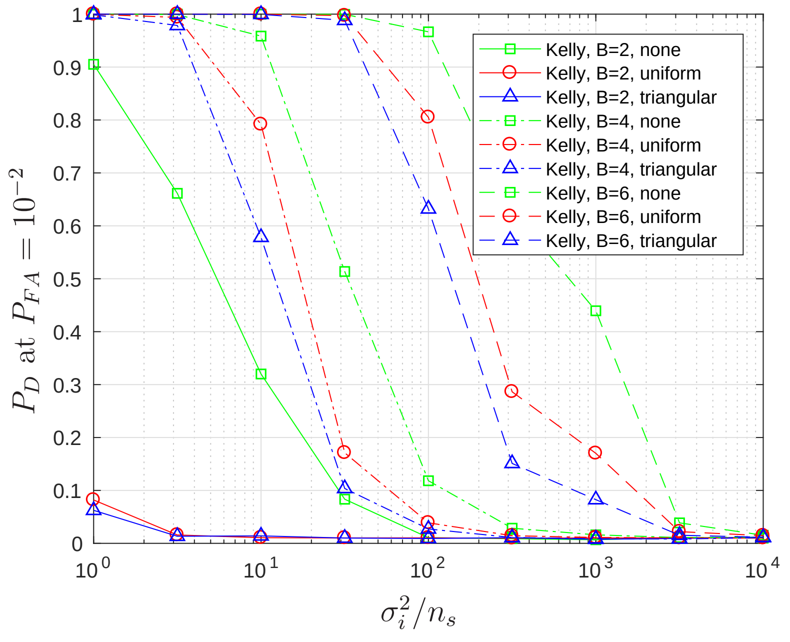


Figure 5.5: Estimated P_D of the Kelly detector vs σ_i^2/n_s for various dither signals, where Δ is set so overload occurs only a small percentage of the time. In this experiment $n_r = 16$, $n_s = 100$, $n_i = 1$, $P_{FA} = 10^{-2}$, and $\sigma_w^2/n_s = 10^{-2}$.

Then, in Figure 5.6, we plot the estimated P_D of the Kelly detector vs σ_i^2/n_s for various dither distributions when Δ is set to maximize the P_D. In this Figure we use the same parameters as in Figure 5.5. From this figure we observe that the detection performance with dithering is still worse than the detection performance without dithering.

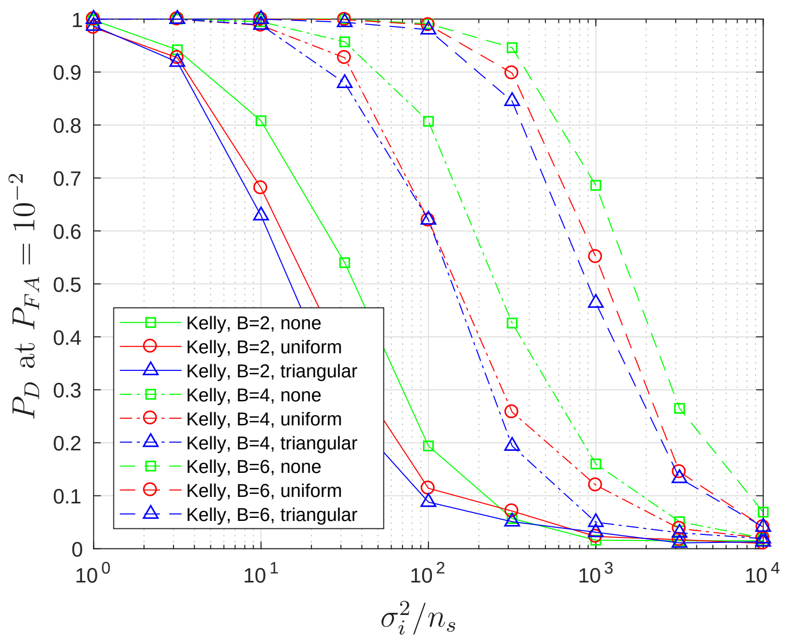


Figure 5.6: Estimated P_D of the Kelly detector vs σ_i^2/n_s for various dither signals, where Δ is set to maximize P_D. In this experiment $n_r = 16$, $n_s = 100$, $n_i = 1$, $P_{FA} = 10^{-2}$, and $\sigma_w^2/n_s = 10^{-2}$.

5.3.2 Detection Performance with Non-uniform Quantization

Next, we test detection performance with non-uniform quantization (companding). Companding increases the dynamic range of the quantizer by using relatively precise quantization for values near 0, while allocating fewer quantization bins for large signals. Companding can be represented mathematically by

$$y = f^{-1}\mathcal{Q}_{B,\Delta}\Big(f(x)\Big), \tag{5.16}$$

where $\mathcal{Q}_{B,\Delta}$ is still the uniform quantization function given in (5.2), and f is the compressor function, and its inverse f^{-1} is the expander function (hence the name companding, which is a morphing of compress and expand).

Two common choices of f are given by μ-law and A-law companding. They are nearly identical, so for simplicity we only considered μ-law. In μ-law companding

$$f(x) = \operatorname{sgn}(x)\frac{\log(1+\mu|x|)}{\log(1+\mu)}, \tag{5.17}$$

where $\mu > 0$ is a tuning parameter.

In Figure 5.7 we plot the estimated P_D of the Kelly detector vs σ_i^2/n_s with and without μ-law companding. In this experiment, $B \in \{4, 6, 8\}$, $n_r = 16$, $n_s = 100$, $n_i = 1$, $\sigma_w^2/n_s = 10^{-2}$, and $P_{FA} = 10^{-2}$. In all cases, Δ (and μ when companding was applied) were optimized to maximize P_D. From this Figure, we see a slight increase in performance when companding was used, but even with companding, detection with any level of quantization under test was eventually killed by the large interference.

5.4 The GLRT with the Quantized Model

In the previous section we saw that the performance of the Kelly detector with quantized measurements was not significantly affected by either dithering or companding. In this section we develop the GLRT for the problem in (5.1) that includes the quantization with the goal of achieving improved detection performance, particularly in the case of strong interference.

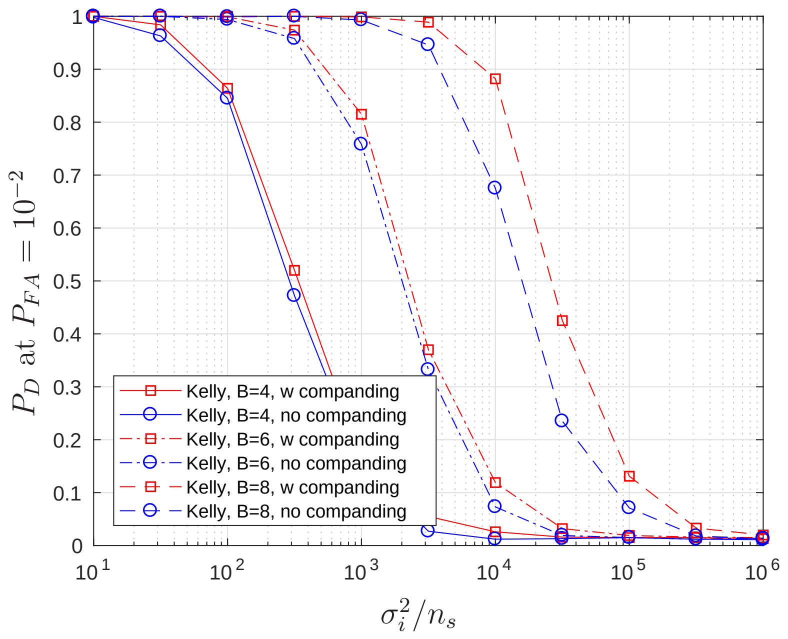


Figure 5.7: Estimated P_D of the Kelly detector vs σ_i^2/n_s with and without companding. In this experiment $n_r = 16$, $n_s = 100$, $n_i = 1$, $P_{FA} = 10^{-2}$, and $\sigma_w^2/n_s = 10^{-2}$.

We follow the model used by McWhorter, where we assume $\boldsymbol{h}$, $\boldsymbol{G}$, and $\boldsymbol{\Psi}$ are deterministic but unknown, $\boldsymbol{W}$ is AWGN with variance σ_w^2, and that n_i is known. We use this model due to the relatively simple form the GLRT will take. With these assumptions the GLRT is

$$\frac{\max_{\boldsymbol{h},\boldsymbol{G},\boldsymbol{\Psi},\sigma_w^2} p(\boldsymbol{Y}|\mathcal{H}_1; \boldsymbol{h}, \boldsymbol{G}, \boldsymbol{\Psi}, \sigma_w^2)}{\max_{\boldsymbol{G},\boldsymbol{\Psi},\sigma_w^2} p(\boldsymbol{Y}|\mathcal{H}_0; \boldsymbol{G}, \boldsymbol{\Psi}, \sigma_w^2)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau. \tag{5.18}$$

For simplicity, we consider real-valued measurements. We begin with the 1-bit case, and later we generalize to the multi-bit case.

5.4.1 The GLRT in the 1-bit Case

In the 1-bit case

$$p(\boldsymbol{Y}|\mathcal{H}_1; \boldsymbol{h}, \boldsymbol{G}, \boldsymbol{\Psi}, \sigma_w^2) = \prod_{i=1:n_r, j=1:n_s} p(y_{ij}|z_{ij}^{(1)}, \sigma_w^2) \tag{5.19}$$

$$= \prod_{i=1:n_r, j=1:n_s} \Phi\bigg(\frac{y_{ij} z_{ij}^{(1)}}{\sigma_w}\bigg), \tag{5.20}$$

where $z_{ij}^{(1)} \triangleq \Big[[\boldsymbol{h}, \boldsymbol{G}][\boldsymbol{s}, \boldsymbol{\Psi}]^{\mathsf{T}}\Big]_{ij}$, Φ is the standard normal CDF, and recall from (5.2) that $y_{ij} \in \{-1, 1\}$. $p(\boldsymbol{Y}|\mathcal{H}_0; \boldsymbol{G}, \boldsymbol{\Psi}, \sigma_w^2)$ is similar to that in (5.19)-(5.20), except $z_{ij}^{(1)}$ is replaced with $z_{ij}^{(0)} \triangleq \Big[\boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{T}}\Big]_{ij}$. The parameter n_i, which manifests as the number of columns of $\boldsymbol{G}$ and $\boldsymbol{\Psi}$, cannot be included in the GLRT because both the numerator and denominator of (5.18) increase with n_i.

The form of (5.20) is calculated from the following. Assuming $y_{ij} = \operatorname{sgn}(z_{ij} + w_{ij})$, where $w_{ij} \sim \mathcal{N}(0, \sigma_w^2)$, then $\Pr\{y_{ij} = 1|z_{ij}\} = \Pr\{z_{ij} + w_{ij} > 0\} = \Pr\{w_{ij} > -z_{ij}\} = 1 - \Phi\Big(-\frac{z_{ij}}{\sigma_w}\Big) = \Phi\Big(\frac{z_{ij}}{\sigma_w}\Big)$, since $\Phi(x) = 1 - \Phi(-x)$. Then, $\Pr\{y_{ij} = -1|z_{ij}\} = 1 - \Pr\{y_{ij} = 1|z_{ij}\} = 1 - \Phi\Big(\frac{z_{ij}}{\sigma_w}\Big) = \Phi\Big(-\frac{z_{ij}}{\sigma_w}\Big)$. Finally, we can write $p(y_{ij}|z_{ij})$ compactly as $\Phi\Big(\frac{y_{ij} z_{ij}}{\sigma_w}\Big)$.

Binary PCA

The optimization problem in the denominator of (5.18) can be recognized as binary principal components analysis (PCA) [76, 77], while the optimization problem in the numerator of (5.18) is a constrained version of binary PCA. In binary PCA, one solves

$$\min_{\boldsymbol{Z}} L(\boldsymbol{Z}; \sigma_w^2) \text{ s.t. } \operatorname{rank} \boldsymbol{Z} \le N, \tag{5.21}$$

where N is the maximum desired rank of $\widehat{\boldsymbol{Z}}$ and

$$L(\boldsymbol{Z}; \sigma_w^2) = \sum_{i=1:n_r, j=1:n_s} -\log \Phi\bigg(\frac{y_{ij} z_{ij}}{\sigma_w}\bigg), \tag{5.22}$$

which is the negative log-likelihood of the model in (5.20). Note that the optimal value of the objective function in (5.21) is independent of σ_w, and therefore σ_w can be assumed to equal 1.

Since the size of $\boldsymbol{Z}$ may be quite large, it is common to work with the factorization $\boldsymbol{Z} = \boldsymbol{L}\boldsymbol{R}^{\mathsf{T}}$ and solve

$$\min_{\boldsymbol{L},\boldsymbol{R}} L(\boldsymbol{L}\boldsymbol{R}^{\mathsf{T}}; \sigma_w^2), \tag{5.23}$$

where $\boldsymbol{L}$ and $\boldsymbol{R}$ are chosen to be of size $n_r \times N$ and $n_s \times N$, respectively and so the total number of optimization variables is $N(n_r + n_s)$, which is smaller than $n_r \times n_s$ for small enough N. Note that Problems (5.21) and (5.23) are both non-convex.

Then, to apply binary PCA to the GLRT in (5.18) with a rank constraint of n_i, the denominator of (5.18) is solved via

$$\min_{\boldsymbol{G},\boldsymbol{\Psi}} L(\boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{T}}; \sigma_w^2), \tag{5.24}$$

while the numerator is solved via

$$\min_{\boldsymbol{h},\boldsymbol{G},\boldsymbol{\Psi}} L([\boldsymbol{h}, \boldsymbol{G}][\boldsymbol{s}, \boldsymbol{\Psi}]^{\mathsf{T}}; \sigma_w^2), \tag{5.25}$$

where in both (5.24) and (5.25), $\boldsymbol{G}$ and $\boldsymbol{\Psi}$ have n_i columns. In the sequel, we will discuss Problem (5.23) in general, where we use the notation $\boldsymbol{L}$ and $\boldsymbol{R}$, which each have N columns.

Regularization

Problems (5.21) and (5.23) are non-convex, and therefore difficult to solve. However, there is a convex relaxation of Problem (5.21) that leads to a regularized form of Problems (5.24) and (5.25), which in practice may be easier to solve.

The convex relaxation of Problem (5.21) is to replace the rank constraint with a nuclear-norm penalty on $\boldsymbol{Z}$, in which case the problem is now

$$\min_{\boldsymbol{Z}} L(\boldsymbol{Z}; \sigma_w^2) + \gamma \|\boldsymbol{Z}\|_*, \tag{5.26}$$

where γ is a tuning parameter.

Problem (5.26) is attractive compared to (5.21) because it is convex, however, in its current form it is difficult to solve numerically due to the large number of elements in $\boldsymbol{Z}$. To deal with numerical difficulties we consider the Frobenius norm regularized problem

$$\min_{\boldsymbol{L},\boldsymbol{R}} L(\boldsymbol{L}\boldsymbol{R}^{\mathsf{T}}; \sigma_w^2) + \frac{\gamma}{2}\Big(\|\boldsymbol{L}\|_F^2 + \|\boldsymbol{R}\|_F^2\Big). \tag{5.27}$$

We use this alternate form because it can be shown [77] that if $(\boldsymbol{L}^*, \boldsymbol{R}^*)$ are rank N and solve (5.27), then $\boldsymbol{Z}^* = \boldsymbol{L}^*\boldsymbol{R}^{*\mathsf{T}}$ solves

$$\min_{\boldsymbol{Z}} L(\boldsymbol{Z}; \sigma_w^2) + \gamma \|\boldsymbol{Z}\|_* \text{ s.t. } \operatorname{rank} \boldsymbol{Z} \leq N. \tag{5.28}$$

The regularization in (5.27) can be interpreted as MAP estimation of $\boldsymbol{L}$ and $\boldsymbol{R}$ under a Gaussian prior. Indeed, if there is an independent Gaussian prior on each element of $\boldsymbol{L}$ and $\boldsymbol{R}$, with zero-mean and variances σ_l^2 and σ_r^2, respectively, then the optimization problem is

$$\min_{\boldsymbol{L},\boldsymbol{R}} L(\boldsymbol{L}\boldsymbol{R}^{\mathsf{T}}; \sigma_w^2) + \frac{1}{2\sigma_l^2}\|\boldsymbol{L}\|_F^2 + \frac{1}{2\sigma_r^2}\|\boldsymbol{R}\|_F^2. \tag{5.29}$$

Problem (5.29) depends on the parameters σ_l^2 and σ_r^2. An equivalent problem is now presented that reduces this to a single parameter. Note that if we scale $\boldsymbol{L}$ by α and $\boldsymbol{R}$ by $1/\alpha$, and the value of $L(\cdot)$ does not change. We will substitute $\widetilde{\boldsymbol{L}} = \sqrt{\frac{\sigma_r}{\sigma_l}}\boldsymbol{L}$

and $\widetilde{\boldsymbol{R}} = \sqrt{\frac{\sigma_l}{\sigma_r}}\boldsymbol{R}$ in (5.29), yielding

$$= \underset{\widetilde{\boldsymbol{L}}, \widetilde{\boldsymbol{R}}}{\arg\min}\, L(\widetilde{\boldsymbol{L}}\widetilde{\boldsymbol{R}}^{\mathsf{T}}; \sigma_w^2) + \frac{1}{2}\frac{1}{\sigma_l \sigma_r}\|\widetilde{\boldsymbol{L}}\|_F^2 + \frac{1}{2}\frac{1}{\sigma_l \sigma_r}\|\widetilde{\boldsymbol{R}}\|_F^2 \tag{5.30}$$

$$= \underset{\widetilde{\boldsymbol{L}}, \widetilde{\boldsymbol{R}}}{\arg\min}\, L(\widetilde{\boldsymbol{L}}\widetilde{\boldsymbol{R}}^{\mathsf{T}}; \sigma_w^2) + \frac{\gamma}{2}\Big(\|\widetilde{\boldsymbol{l}}\|_F^2 + \|\widetilde{\boldsymbol{R}}\|_F^2\Big), \tag{5.31}$$

where $\gamma = \frac{1}{\sigma_r \sigma_l}$, which matches the problem in (5.27). Finally, note that $L(\boldsymbol{Z}; \sigma_w^2)$ depends on σ_w^2. If we want to fix $\sigma_w^2 = 1$ in the optimization (as is common with PCA), we must scale $\widetilde{\boldsymbol{L}}$ and $\widetilde{\boldsymbol{R}}$ each by $\sqrt{\sigma_w}$, in which case

$$\gamma = \frac{\sigma_w}{\sigma_l \sigma_r}. \tag{5.32}$$

Algorithms for Binary PCA

Here we describe several approaches from the literature to solving Problem (5.27). The first, given by Collins [76], is a coordinate descent approach that loops over elements in $\boldsymbol{L}$ and $\boldsymbol{R}$ and minimizes the objective in (5.27) wrt each scalar variable. This process is then repeated over a number of epochs. Another approach that is given in [78] attempts to solve Problem (5.27) via majorization-minimization. This particular approach produces a non-increasing sequence of loss function values. A final approach given in [77] alternates between gradient steps wrt $\boldsymbol{L}$ and $\boldsymbol{R}$, while keeping track of separate learning rates for each row of $\boldsymbol{L}$ and $\boldsymbol{R}$. Regarding initialization of $\boldsymbol{L}$ and $\boldsymbol{R}$, all methods that have been mentioned can be initialized using the SVD of quantized $\boldsymbol{Y}$. This initialization typically converges to a better final value of the objective than a random initialization [77]. In our experiments, we use the alternating gradient descent approach in [77] combined with the SVD initialization method.

Gradient

Here we calculate the gradient wrt $\boldsymbol{L}$ and $\boldsymbol{R}$ of the objective in (5.23), which is required for the alternating gradient descent algorithm in [77]. We then show a novel quadratic approximation that improves the numerical robustness of evaluating the gradient.

Writing

$$L(\boldsymbol{L}\boldsymbol{R}^\mathsf{T};\sigma_w^2) = -\sum_{i=1:n_r,j=1:n_s} \log \Phi\Bigg(\underbrace{\frac{y_{ij}\sum_{n=1}^{N} l_{in} r_{jn}}{\sigma_w}}_{\triangleq a_{ij}}\Bigg), \tag{5.33}$$

we can calculate

$$\frac{\mathrm{d}}{\mathrm{d}l_{in}} = -\sum_j \underbrace{\frac{\phi(a_{ij})y_{ij}}{\Phi(a_{ij})\sigma_w}}_{\triangleq f_{ij}} r_{jn} \tag{5.34}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}r_{jn}} = -\sum_i f_{ij} l_{in}. \tag{5.35}$$

Using (5.34) and (5.35), we can form

$$\nabla_{\boldsymbol{L}} L(\boldsymbol{L}\boldsymbol{R}^\mathsf{T};\sigma_w^2) = -\boldsymbol{F}\boldsymbol{R} \tag{5.36}$$

and

$$\nabla_{\boldsymbol{R}} L(\boldsymbol{L}\boldsymbol{R}^\mathsf{T};\sigma_w^2) = -\boldsymbol{F}^\mathsf{T}\boldsymbol{L}. \tag{5.37}$$

Quadratic Approximation for Numerical Robustness Numerical problems arise in (5.33) when any a_{ij} are small. In order to prevent numerical problems from occurring, we approximate the function $\log\Phi(x)$ with a second-order Taylor series[17]

[17]The Feller approximation [79, 80] gives a more accurate approximation wrt absolute difference, but is not continuous at the point x_0. Continuity of the spliced function and its derivative were highly desired due to the use of gradient-descent algorithms, hence the use of a second order Taylor series approximation.

about the point x_0 for all $x < x_0$. The second-order Taylor series is given by

$$\underbrace{\log\Big(\Phi(x)\Big)}_{\triangleq l(x)} \approx l(x_0) + l'(x_0)(x - x_0) + \frac{l''(x_0)}{2}(x - x_0)^2, \tag{5.38}$$

where $l'(x_0) = \frac{\phi(x_0)}{\Phi(x_0)}$ and $l''(x_0) = \frac{-x_0\phi(x_0)\Phi(x_0) - \phi(x_0)^2}{\Phi(x_0)^2}$ (where the latter uses the fact that $\phi'(x_0) = -x_0\phi(x_0)$). Then to calculate the gradient with the quadratic approximation, replace f_{ij} for all (i, j) in (5.36) and (5.37) where $a_{ij} < x_0$ with

$$f_{ij} \approx \Big(l'(x_0) + l''(x_0)(a_{ij} - x_0)\Big)\frac{y_{ij}}{\sigma_w}. \tag{5.39}$$

In our experiments, we set $x_0 = -5$.

Regularization If the quadratic regularization in (5.27) is used, note that

$$\nabla_{\boldsymbol{X}} \frac{\gamma}{2}\|\boldsymbol{X}\|_F^2 = \gamma \boldsymbol{X}. \tag{5.40}$$

Using this, the gradients of the regularized cost in (5.27) are

$$\nabla_{\boldsymbol{L}}\Bigg(L(\boldsymbol{L}\boldsymbol{R}^{\mathsf{T}}; \sigma_w^2) + \frac{\gamma}{2}\Big(\|\boldsymbol{L}\|_F^2 + \|\boldsymbol{R}\|_F^2\Big)\Bigg) = -\boldsymbol{F}\boldsymbol{R} + \gamma\boldsymbol{L} \tag{5.41}$$

and

$$\nabla_{\boldsymbol{R}}\Bigg(L(\boldsymbol{L}\boldsymbol{R}^{\mathsf{T}}; \sigma_w^2) + \frac{\gamma}{2}\Big(\|\boldsymbol{L}\|_F^2 + \|\boldsymbol{R}\|_F^2\Big)\Bigg) = -\boldsymbol{F}^{\mathsf{T}}\boldsymbol{L} + \gamma\boldsymbol{R}. \tag{5.42}$$

5.4.2 Multi-bit Case

Now we consider the GLRT in the multi-bit case.

Likelihood Function

For a given number of bits B and bin-size Δ used in (5.2), let $\mathcal{B} = \{b_0, ..., b_{2^B}\}$ be an increasing list of numbers that correspond to the boundaries of the quantizer decision regions. In this case, $b_0 = -\infty$, $b_{2^B} = \infty$, and $b_i - b_{i-1} = \Delta$ for all $i \neq 0, 2^B$. For a given y, let $b^-(y) \in \mathcal{B}$ and $b^+(y) \in \mathcal{B}$ correspond to the lower and upper boundaries, respectively, of the decision region that maps to y.

Then, $p(y|z) = \Pr\{z + w \in [b^-(y), b^+(y)]\} = \Phi\Big(\frac{z-b^-(y)}{\sigma_w}\Big) - \Phi\Big(\frac{z-b^+(y)}{\sigma_w}\Big)$ for $w \sim \mathcal{N}(0, \sigma_w^2)$. Using this, the likelihood used in the GLRT in (5.18) is

$$\begin{aligned} p(\boldsymbol{Y}|\boldsymbol{Z}; \sigma_w^2) &= \prod_{i=1:n_r, j=1:n_s} p(y_{ij}|z_{ij}, \sigma_w^2) && (5.43)\\ &= \prod_{i=1:n_r, j=1:n_s} \Phi\bigg(\frac{z_{ij} - b_{ij}^-}{\sigma_w}\bigg) - \Phi\bigg(\frac{z_{ij} - b_{ij}^+}{\sigma_w}\bigg), && (5.44)\end{aligned}$$

where $b_{ij}^- \triangleq b^-(y_{ij})$ (and likewise with $+$) for notational simplicity. Note that in the case $B = 1$, $b_0 = -\infty$, $b_1 = 0$, and $b_2 = \infty$, and (5.44) corresponds to (5.20). Also note that the dependence of $p(\boldsymbol{Y}|\boldsymbol{Z}; \sigma_w^2)$ on $\boldsymbol{Y}$ manifests through the various $b_{ij}^{+/-}$.

Now we will briefly summarize similarities and differences to the 1-bit case. First, the quadratic regularization in (5.27) can still be used, as can the alternating gradient descent algorithm and the SVD initialization. However, in the multi-bit case, the optimal value of the likelihood is not invariant to the choice of σ_w^2, and therefore σ_w^2 must be included as an estimand in the GLRT.

Gradient

Similar to the 1-bit case, we calculate the gradient of $L(\boldsymbol{Z}; \sigma_w^2) \triangleq -\log p(\boldsymbol{Y}|\boldsymbol{Z}, \sigma_w^2)$ from (5.44) wrt the factors $\boldsymbol{Z} = \boldsymbol{L}\boldsymbol{R}^\mathsf{T}$. We then show a novel quadratic approximation that increases the robustness of evaluating the gradient.

Noting that

$$L(\boldsymbol{L}\boldsymbol{R}^\mathsf{T}; \sigma_w^2) = \sum_{ij} -\log\left(\Phi\bigg(\underbrace{\frac{\sum_k l_{ik} r_{jk} - b_{ij}^-}{\sigma_w}}_{\triangleq a_{ij}^-}\bigg) - \Phi\bigg(\underbrace{\frac{\sum_k l_{ik} r_{jk} - b_{ij}^+}{\sigma_w}}_{\triangleq a_{ij}^+}\bigg)\right), \qquad (5.45)$$

$$\frac{\mathrm{d}}{\mathrm{d}l_{ik}}L(\boldsymbol{L}\boldsymbol{R}^{\mathsf{T}};\sigma_w^2) = -\sum_j \underbrace{\frac{\phi(a_{ij}^-) - \phi(a_{ij}^+)}{\Phi(a_{ij}^-) - \Phi(a_{ij}^+)}\frac{1}{\sigma_w}}_{\triangleq f_{ij}} r_{jk} \tag{5.46}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}r_{jk}}L(\boldsymbol{L}\boldsymbol{R}^{\mathsf{T}};\sigma_w^2) = -\sum_i f_{ij} l_{ik}, \tag{5.47}$$

which can be written as

$$\nabla_{\boldsymbol{L}} L(\boldsymbol{L}\boldsymbol{R}^{\mathsf{T}};\sigma_w^2) = -\boldsymbol{F}\boldsymbol{R} \tag{5.48}$$

and

$$\nabla_{\boldsymbol{R}} L(\boldsymbol{L}\boldsymbol{R}^{\mathsf{T}};\sigma_w^2) = -\boldsymbol{F}^{\mathsf{T}}\boldsymbol{L}. \tag{5.49}$$

Quadratic Approximation for Numerical Robustness Similar to the 1-bit case, numerical problems may arise if $\Phi(a_{ij}^-) - \Phi(a_{ij}^+)$ is very close to zero. In order to avoid such problems, we will approximate

$$l_{ij}(x) \triangleq \log\left(\Phi\left(\frac{x - b_{ij}^-}{\sigma_w}\right) - \Phi\left(\frac{x - b_{ij}^+}{\sigma_w}\right)\right) \tag{5.50}$$

with a second-order Taylor series about the point $x = x_0$ whenever x is in a range of values that cause numerical problems.

First, we note that a second-order Taylor series is a particularly good approximation to (5.50) when there is no clipping (quantizer overload). Let $\widetilde{x} = x - b_{ij}^+$. Then after substitution,

$$l_{ij}(\widetilde{x}) = \log\left(\Phi\left(\frac{\widetilde{x} + b_{ij}^+ - b_{ij}^-}{\sigma_w}\right) - \Phi\left(\frac{\widetilde{x}}{\sigma_w}\right)\right). \tag{5.51}$$

Since there is no clipping, $b_{ij}^+ - b_{ij}^- = \Delta$. Then after substitution, and for small Δ,

$$\log\left(\Phi\left(\frac{\widetilde{x} + \Delta}{\sigma_w}\right) - \Phi\left(\frac{\widetilde{x}}{\sigma_w}\right)\right) \approx \log\left(\frac{\Delta}{\sigma_w}\phi\left(\frac{\widetilde{x}}{\sigma_w}\right)\right) \tag{5.52}$$

$$\approx c_1\widetilde{x}^2 + c_2 \tag{5.53}$$

for constants c_1 and c_2.

The second order Taylor series of $l_{ij}(x)$ about x_0 (after dropping the ij subscripts for simplicity) is given by

$$l(x) \approx l(x_0) + l'(x_0)(x - x_0) + \frac{1}{2}l''(x_0)(x - x_0)^2, \tag{5.54}$$

where

$$l'(x) = \frac{\frac{1}{\sigma_w}\left(\phi\left(\frac{x-b^-}{\sigma_w}\right) - \phi\left(\frac{x-b^+}{\sigma_w}\right)\right)}{\Phi\left(\frac{x-b^-}{\sigma_w}\right) - \Phi\left(\frac{x-b^+}{\sigma_w}\right)} \tag{5.55}$$

and

$$l''(x) = \frac{-\left(\phi\left(\overline{x}^-\right)\left(\overline{x}^-\right) - \phi\left(\overline{x}^+\right)\left(\overline{x}^+\right)\right)\left(\Phi\left(\overline{x}^-\right) - \Phi\left(\overline{x}^+\right)\right) - \left(\phi\left(\overline{x}^-\right) - \phi\left(\overline{x}^+\right)\right)^2}{\sigma_w^2\left(\Phi\left(\overline{x}^-\right) - \Phi\left(\overline{x}^+\right)\right)^2}, \tag{5.56}$$

where $\overline{x}^+ \triangleq \frac{x-b^+}{\sigma_w}$ and $\overline{x}^- \triangleq \frac{x-b^-}{\sigma_w}$.

In this case where $b^- = -\infty$,

$$l(x) = \log\left(1 - \Phi\left(\frac{x - b^+}{\sigma_w}\right)\right), \tag{5.57}$$

$$l'(x) = \frac{-\frac{1}{\sigma_w}\phi\left(\frac{x-b^+}{\sigma_w}\right)}{1 - \Phi\left(\frac{x-b^+}{\sigma_w}\right)}, \tag{5.58}$$

and

$$l''(x) = \frac{\phi\left(\frac{x-b^+}{\sigma_w}\right)\left(\frac{x-b^+}{\sigma_w}\right)\left(1 - \Phi\left(\frac{x-b^+}{\sigma_w}\right)\right) - \phi\left(\frac{x-b^+}{\sigma_w}\right)^2}{\sigma_w^2\left(1 - \Phi\left(\frac{x-b^+}{\sigma_w}\right)\right)^2}. \tag{5.59}$$

In the case where $b^+ = \infty$,

$$l(x) = \log\Phi\left(\frac{x - b_{ij}^-}{\sigma_w}\right), \tag{5.60}$$

$$l'(x) = \frac{\frac{1}{\sigma_w^2}\phi\left(\frac{x-b^-}{\sigma_w}\right)}{\Phi\left(\frac{x-b^-}{\sigma_w}\right)}, \tag{5.61}$$

and

$$l''(x) = \frac{-\phi\left(\frac{x-b^-}{\sigma_w}\right)\left(\frac{x-b^-}{\sigma_w}\right)\Phi\left(\frac{x-b^-}{\sigma_w}\right) - \phi\left(\frac{x-b^-}{\sigma_w}\right)^2}{\sigma_w^2\Phi\left(\frac{x-b^-}{\sigma_w}\right)^2}. \tag{5.62}$$

Unlike the 1-bit case, it is more difficult to choose when to apply the Taylor series, i.e., what vales of x trigger using the Taylor series approximation. It is also not clear how to best choose x_0. If Δ is small, the option we recommend is to apply the Taylor series approximation for all x, and to set $x_0 = y_{ij}$.

5.4.3 Summary of the GLRT

Putting everything together, to evaluate the GLRT in (5.18) with a fixed n_i, we first solve

$$\left\{\widehat{\boldsymbol{G}}^{(0)}, \widehat{\boldsymbol{\Psi}}^{(0)}\right\} = \underset{\boldsymbol{G},\boldsymbol{\Psi}}{\arg\min}\, L(\boldsymbol{G}\boldsymbol{\Psi}^{\mathsf{T}}; \sigma_w^2) + \frac{\gamma}{2}\left(\|\boldsymbol{G}\|_F^2 + \|\boldsymbol{\Psi}\|_F^2\right) \tag{5.63}$$

and

$$\left\{\widehat{\boldsymbol{h}}, \widehat{\boldsymbol{G}}^{(1)}, \widehat{\boldsymbol{\Psi}}^{(1)}\right\} = \underset{\boldsymbol{h},\boldsymbol{G},\boldsymbol{\Psi}}{\arg\min}\, L([\boldsymbol{h}, \boldsymbol{G}][\boldsymbol{s}, \boldsymbol{\Psi}]^{\mathsf{T}}; \sigma_w^2) + \frac{\gamma}{2}\left(\|[\boldsymbol{h}, \boldsymbol{G}]\|_F^2 + \|[\boldsymbol{s}, \boldsymbol{\Psi}]\|_F^2\right), \tag{5.64}$$

where in both cases $\boldsymbol{G}$ and $\boldsymbol{\Psi}$ have n_i columns and we assumed $\sigma_w^2 = 1$. Then, we evaluate

$$\frac{p(\boldsymbol{Y}|\mathcal{H}_1; \widehat{\boldsymbol{h}}, \widehat{\boldsymbol{G}}^{(1)}, \widehat{\boldsymbol{\Psi}}^{(1)}, \sigma_w^2)}{p(\boldsymbol{Y}|\mathcal{H}_0; \widehat{\boldsymbol{G}}^{(0)}, \widehat{\boldsymbol{\Psi}}^{(0)}, \sigma_w^2)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau, \tag{5.65}$$

where $\sigma_w^2 = 1$ in both the numerator and denominator of (5.65).

5.4.4 Numerical Results

In this section we evaluate the performance of the quantized GLRT. First, in Figure 5.8, we plot the estimated P_D of various detectors vs σ_i^2/n_s in the one-bit case with real-valued measurements. Included in our test are the Kelly, KMR, and McWhorter unquantized detectors, as well as the quantized GLRT given in equations (5.63)-(5.65) (referred to as “q-GLRT” in the legend). For simplicity, in this test, the true n_i was provided to all algorithms that required it, and the γ used by PCA was calculated via (5.32), where the appropriate variances were provided by an oracle. In this experiment, $n_r = 16$, $n_s = 100$, $n_i = 1$, $\sigma_w^2/n_s = 10^{-2}$, and $P_{FA} = 10^{-2}$. From this figure, we observe that the three low-rank detectors (KMR, McWhorter, and the quantized GLRT) have approximately the same performance, and are out-performing the Kelly detector. Regardless, all detectors under test eventually fail under large interference.

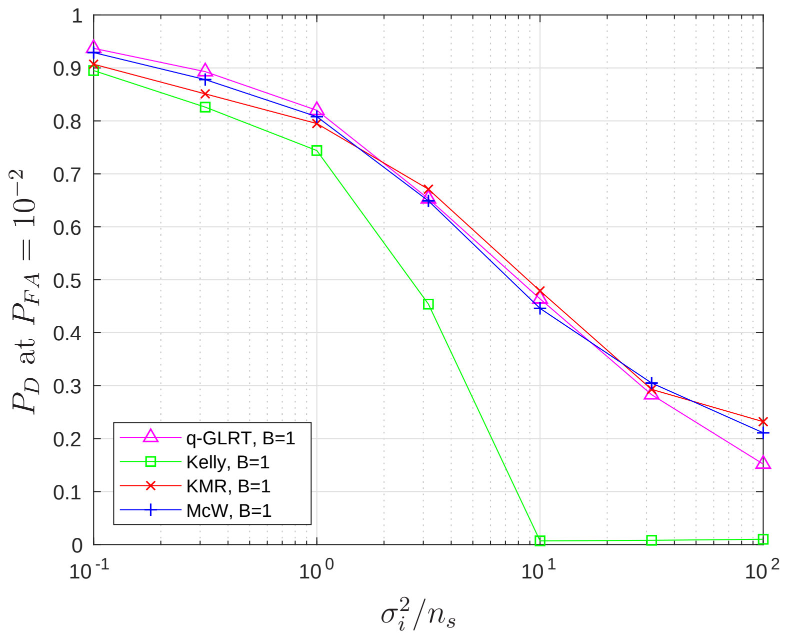


Figure 5.8: Estimated P_D of the various detectors in the single-bit case vs σ_i^2/n_s. In this experiment $n_r = 16$, $n_s = 100$, $n_i = 1$, $P_{FA} = 10^{-2}$, and $\sigma_w^2/n_s = 10^{-2}$.

Then, in Figure 5.9, we examine performance as B is increased. We plot empirical receiver operating characteristic (ROC) curves (which show P_D as a function of P_{FA}) for various detectors and $B \in \{1, 2, 3, \infty\}$. In this experiment $n_r = 16$, $n_s = 100$, $n_i = 1$, $\sigma_w^2/n_s = 10^{-2}$, and $\sigma_i^2/n_s = 1$. As before, the true value of n_i was provided to all detectors that required it, and the γ used by PCA was again calculated via (5.32), where the appropriate variances were provided by an oracle. Note that the quantized GLRT cannot be evaluated when $B = \infty$. From this figure, we see that performance for all detectors improves as B increases. The relative performance between the detectors is Kelly is the worst, followed by the KMR detector, and finally the quantized GLRT and McWhorter are approximately tied for the best.

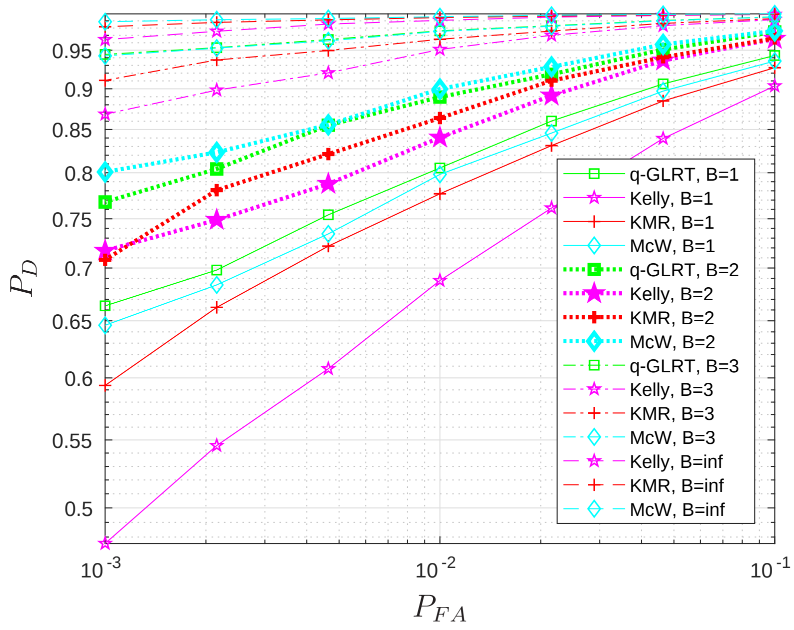


Figure 5.9: Estimated ROC curves of various detectors for different values of B. In this experiment $n_r = 16$, $n_s = 100$, $n_i = 1$, $\sigma_i^2/n_s = 1$, and $\sigma_w^2/n_s = 10^{-2}$.

5.4.5 Summary

In this section we developed the GLRT for the quantized model in the case of real-valued measurements. Based on the assumptions we made regarding $\boldsymbol{G}$, $\boldsymbol{\Psi}$, and $\boldsymbol{W}$, the two estimation problems in the GLRT involve PCA from quantized measurements, which is a particular example of generalized bilinear inference. While we admit there is future work that could be performed in this area, including: extension to the complex case, incorporating optimization of σ_w^2 in the multi-bit case, and applying other generalized low-rank inference algorithms such as low-rank AMP (LowRAMP) [7] or Bilinear GAMP (BiGAMP) [6], with our work so far there is not evidence that it will improve detection significantly in the high interference case, which is the regime of interest (although the PCA-based GLRT may offer a slight improvement over the Kelly detector). Therefore, in the next section we again turn to alternate techniques to improve detection performance in the case of high interference.

5.5 Pre-processing Techniques

So far, we have observed the general trend that detection from quantized measurement performs adequately well when the interference is small, but suffers from a lack of dynamic range when the interference is large. Techniques such as dithering, companding, or the GLRT with the quantized model have been of limited value, and so in this section we explore alternative ways of quantizing the signal.

We note the various approaches we propose in the sequel require a significant change of the receiver architecture. Some methods require a matrix multiplication of the received signal prior to quantization, which may not be feasible in practice, but we consider them anyway as a point of comparison. In light of this, we also test more "implementable" approaches, where the matrix multiplication consists of unit-modulus elements (i.e., the only operations are phase-shifting and adding), which can be implemented more easily in the analog domain.

5.5.1 Beamforming

We first consider beamforming. In beamforming, at each time snapshot the received signal from each antenna are linearly combined via beamforming vector $\boldsymbol{b} \in \mathbb{C}^{n_r}$, prior to quantization. This can be written as

$$\boldsymbol{y} = \mathcal{Q}_{B,\Delta}\Big(\boldsymbol{U}^{\mathsf{H}}\boldsymbol{b}\Big) \in \mathbb{C}^{n_s}, \tag{5.66}$$

where the detection statistic is computed from $\boldsymbol{y}$ instead of $\boldsymbol{U}$ or $\mathcal{Q}_{B,\Delta}(\boldsymbol{U})$. Note that

$$\boldsymbol{U}^{\mathsf{H}}\boldsymbol{b} = \boldsymbol{s}\boldsymbol{h}^{\mathsf{H}}\boldsymbol{b} + \boldsymbol{\Psi}\boldsymbol{G}^{\mathsf{H}}\boldsymbol{b} + \boldsymbol{W}^{\mathsf{H}}\boldsymbol{b}, \tag{5.67}$$

which, under the assumption of white, Gaussian $\boldsymbol{\Psi}$ and $\boldsymbol{W}$, is equivalent to

$$\boldsymbol{U}^{\mathsf{H}}\boldsymbol{b} = h\boldsymbol{s} + \boldsymbol{n}, \tag{5.68}$$

where h is an unknown, complex gain and $\boldsymbol{n}$ is AWGN with unknown variance σ_n^2. The appropriate GLRT to apply to (5.68) is therefore

$$\frac{\max_{h,\sigma_n^2} p(\boldsymbol{y}|\mathcal{H}_1; h, \sigma_n^2)}{\max_{\sigma_n^2} p(\boldsymbol{y}|\mathcal{H}_0; \sigma_n^2)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau, \tag{5.69}$$

which simplifies to

$$\frac{\|\boldsymbol{y}\|^2}{\|\boldsymbol{y}\boldsymbol{P}_{\boldsymbol{s}}^{\perp}\|^2} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \tau. \tag{5.70}$$

Now we discuss various approaches to designing $\boldsymbol{b}$. As mentioned before, we consider approaches with and without regard to practicality of the implementation.

The first method is the well known Capon beamformer, which maximizes the signal to interference plus noise ratio (SINR). Given $\boldsymbol{h}$ and $\boldsymbol{\Sigma} \triangleq \text{Cov}\left\{\boldsymbol{Y}\right\}$,

$$\boldsymbol{b}_{\mathsf{capon}} = \boldsymbol{\Sigma}^{-1}\boldsymbol{h}. \tag{5.71}$$

Then, with implementation in mind, Smith [81] provides a conjugate-gradient algorithm that, given $\boldsymbol{h}$ and $\boldsymbol{\Sigma}$, yields the "Capon-phase" beamformer $\boldsymbol{b}_{\mathsf{capon\text{-}phase}}$ that maximizes the SINR after beamforming while restricting elements in $\boldsymbol{b}_{\mathsf{capon\text{-}phase}}$ to unit-modulus.

Now we test the performance of these detection methods. In Figure 5.10 we plot the estimated P_D vs σ_i^2/n_s for a fixed $P_{FA} = 10^{-2}$. In this test, $n_r = 16$, $n_s = 100$, $n_i = 1$, and $\sigma_w^2/n_s = 10^{-2}$. Included in our test is the detector in (5.70) applied to the output of (5.66) in the case of Capon beamforming and "Capon-phase" beamforming, and also the Kelly detector applied to the quantized measurements without beamforming. In this test, $B = 3$, and as usual Δ was chosen for each approach to maximize the estimated P_D. The Capon and Capon-phase beamformers used ML estimates of $\boldsymbol{h}$ and $\boldsymbol{\Sigma}$, computed from $\boldsymbol{U}$ generated under $\mathcal{H}_1$. In this figure, the Capon approach worked perfectly for all levels of interference. The Capon-phase approach worked well, but eventually started to degrade for high levels of interference.

5.5.2 Pre-Whitening

In this section we consider approaches based on estimating $\boldsymbol{\Sigma}$ from quantized measurements and adjusting the pre-processing accordingly. In particular, if we can estimate $\boldsymbol{\Sigma}$, we would like to pre-whiten[18] the signal prior to quantization. We propose an iterative method that updates the whitening operation using the estimated

[18]For reference, if $\text{Cov}\{\boldsymbol{u}\} = \boldsymbol{\Sigma}$, then $\text{Cov}\{\boldsymbol{R}\boldsymbol{u}\} = \boldsymbol{I}$ if $\boldsymbol{R}^{\mathsf{H}}\boldsymbol{R} = \boldsymbol{\Sigma}^{-1}$. Here $\boldsymbol{R}$ has no relation to the $\boldsymbol{R}$ in Section 5.4.

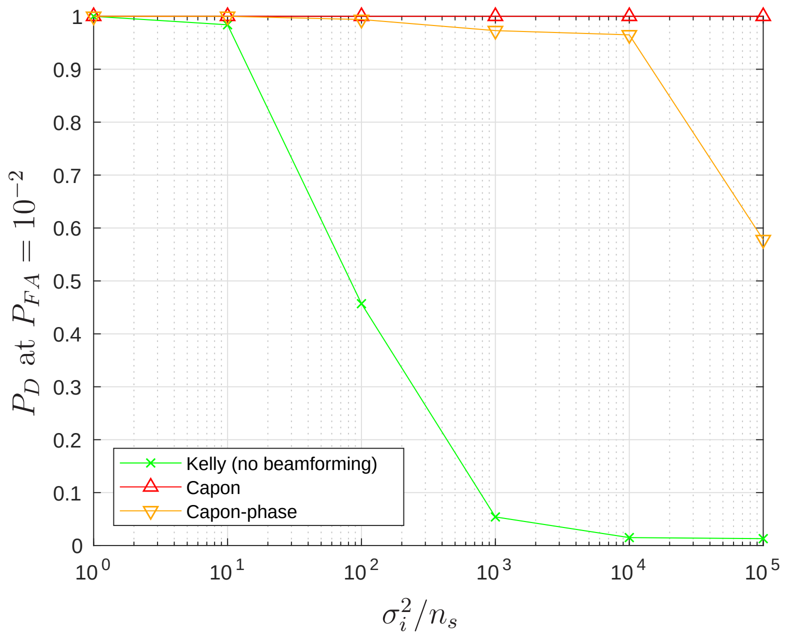


Figure 5.10: P_D vs σ_i^2/n_s with different beamforming techniques.

covariance matrix of the data post-whitening and post-quantization. This work is similar to an approach taken in [82], but the beamformer they design is different, and they do not take an iterative approach to improving beamforming or pre-whitening operation.

Iterative Method to Improve Whitening Matrix

Now we describe our iterative method to estimating the whitening matrix. Each "epoch" takes place later in time, and so the measurements obtained at one epoch are different from the measurements obtained at an earlier epoch. Due to this, the approach assumes the distribution of $\boldsymbol{U}_{(t)}$ does not change wrt t, where t represents the epoch. We begin with the first epoch of quantized measurements $\boldsymbol{Y}_{(1)} = \mathcal{Q}_{B,\Delta}\left(\boldsymbol{U}_{(1)}\right)$, from which we estimate $\widehat{\boldsymbol{\Sigma}}_{(1)} = \boldsymbol{Y}_{(1)}\boldsymbol{Y}_{(1)}^{\mathsf{H}}/n_s$. Then, at a second epoch, $\boldsymbol{Y}_{(2)} =$

$\mathcal{Q}_{B,\Delta}\Big(\boldsymbol{R}_{(1)}\boldsymbol{U}_{(2)}\Big)$, and we estimate $\widehat{\boldsymbol{\Sigma}}_{(2)} = \boldsymbol{Y}_{(2)}\boldsymbol{Y}_{(2)}^{\mathsf{H}}/n_s$. Then, we can whiten the 3rd epoch via $\boldsymbol{Y}_{(3)} = \mathcal{Q}_{B,\Delta}\Big(\boldsymbol{R}_{(2)}\boldsymbol{R}_{(1)}\boldsymbol{U}_{(3)}\Big)$, and so on, where $\boldsymbol{R}_{(t)}^{\mathsf{H}}\boldsymbol{R}_{(t)} = \widehat{\boldsymbol{\Sigma}}_{(t)}^{-1}$.

In Figure 5.11, we show the estimated detection probability vs σ_i^2/n_s of the Kelly detector applied to quantized measurements without pre-whitening, as well as the Kelly detector applied to pre-whitened data for different values of $t \in \{1, ..., 5\}$ (shown in parentheses). In this experiment, $n_r = 16$, $n_s = 100$, $n_i = 1$, $\sigma_w^2 = 10^{-2}$, $P_{FA} = 10^{-2}$, and $B = 3$. In this simulation, $\boldsymbol{U}_{(t)}$ were different realizations from the same data distribution, generated in the $\mathcal{H}_0$ case. In this experiment we observe an improvement in detection performance with the proposed pre-whitening method. Using a single epoch in the estimate improves the performance most significantly, while marginal improvements in performance diminish with additional epochs. Although, we note that for large enough interference, the pre-whitened method fails for all of the number of epochs under test.

Implementation Details

For a particular $\boldsymbol{\Sigma}$, the whitening matrix $\boldsymbol{R}$ is not unique. Two possible methods of computing $\boldsymbol{R}$ from $\boldsymbol{\Sigma}$ are from the Cholesky and eigenvalue decompositions. The eigenvalue decomposition is more interpretable, and possibly superior from an implementation standpoint. If $\boldsymbol{\Sigma}^{-1} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathsf{H}}$ via the eigenvalue decomposition, then $\boldsymbol{R} = \boldsymbol{\Lambda}^{1/2}\boldsymbol{V}^{\mathsf{H}}$. When $\boldsymbol{R}$ is applied to a measured signal, this has the effect of first rotating the signal via $\boldsymbol{V}^{\mathsf{H}}$ into a "beamspace" representation, and then scaling each of the beams individually via $\boldsymbol{\Lambda}^{1/2}$. If only $t = 1$ epochs are used to estimate the net

$\boldsymbol{R}$, it is possible to implement the $\boldsymbol{\Lambda}^{1/2}$ action by adjusting the gains on the individual ADCs (similar to companding[19]), prior to applying $\boldsymbol{\Lambda}^{1/2}$ to the post-quantized samples, in which case the only thing that needs to be implemented in the analog domain is the rotation $\boldsymbol{V}^{\mathsf{H}}$. Specifically,

$$\mathcal{Q}_{B,\Delta}\Big(\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}^{\mathsf{H}}\boldsymbol{u}\Big) = \boldsymbol{\Lambda}^{1/2}\mathcal{Q}_{B,\boldsymbol{\Lambda}^{-1/2}\Delta}\Big(\boldsymbol{V}^{\mathsf{H}}\boldsymbol{u}\Big), \tag{5.72}$$

where $\mathcal{Q}_{B,\boldsymbol{\Lambda}^{-1/2}\Delta}(\cdot)$ has channel-wise $\Delta_i = \lambda_i^{-1/2}\Delta$. However, if $t > 1$ epochs are used, it is unclear if the action of all $\boldsymbol{\Lambda}_{(t)}^{1/2}$ can be incorporated in the quantizer, because the overall whitening plus quantization operation is

$$\mathcal{Q}_{B,\Delta}\Big(\boldsymbol{\Lambda}_{(t)}^{1/2}\boldsymbol{V}_{(t)}^{\mathsf{H}}...\boldsymbol{\Lambda}_{(1)}^{1/2}\boldsymbol{V}_{(1)}^{\mathsf{H}}\boldsymbol{u}\Big). \tag{5.73}$$

One area of future work is to see if the net whitening operation with an arbitrary number of epochs is able to be written (or approximated) as a single rotation followed by a channel-wise scaling. In the sequel we observe what happens when we restrict $\boldsymbol{V}_{(t)}$ to unit modulus elements, which can then be implemented the the analog domain more efficiently.

Implementation with Unit-modulus Elements

A simple approach to implementing the whitening filter with phase-shifters only is to replace elements in $\boldsymbol{V}$ with $\widetilde{v}_{ij} = v_{ij}/|v_{ij}|$. On top of that, if only discrete phases are allowed, the argument of $\widetilde{v}_{ij}$ can be quantized to one of the allowed phases. A justification for this approach is now given. The rows of $\boldsymbol{V}^{\mathsf{H}}$ indicate the direction of each "beam". We want to replace rows of $\boldsymbol{V}^{\mathsf{H}}$ with vectors that are as parallel

[19]In companding, the same non-uniform quantization function is used on every ADC, whereas here each ADC uses a uniform quantization function but with ADC-dependent Δ.

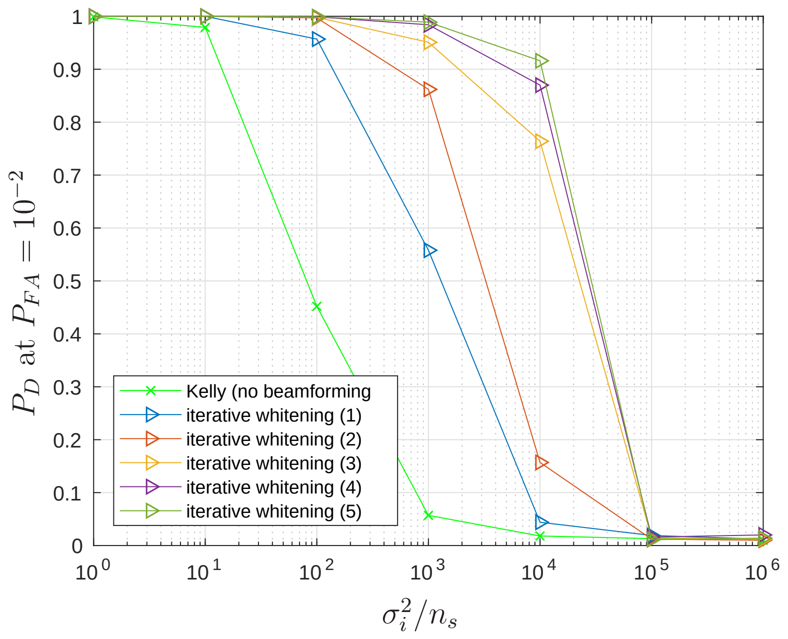


Figure 5.11: P_D vs σ_i^2/n_s with iterative whitening technique. Number in parentheses shows number of iterations (epochs).

as possible while being restricted to unit-modulus elements. This problem can be written as

$$\widetilde{\boldsymbol{v}}_i = \arg\max_{\boldsymbol{v}} \boldsymbol{v}_i^{\mathsf{H}} \boldsymbol{v} \text{ s.t. } |v_j| = 1 \forall j, \tag{5.74}$$

where $\boldsymbol{v}_i^{\mathsf{H}}$ is the ith row of $\boldsymbol{V}^{\mathsf{H}}$. From this problem, it is clear the solution is $\widetilde{v}_{ij} = v_{ij}/|v_{ij}|$ for all i and j.

In Figures 5.12 and 5.13, we evaluate the performance of the proposed iterative whitening method when elements in $\boldsymbol{V}$ are restricted to unit modulus and unit modulus with discrete phases, respectively. We used the same parameters as in the experiment in Figure 5.11. We observe a moderate performance loss when comparing to the phase only performance in Figure 5.12 to the arbitrary $\boldsymbol{V}$ performance in Figure 5.11. Then, there is a small performance loss when comparing the phase only performance in Figure 5.12 to the discrete phase only performance in Figure 5.13. Overall, in Figure 5.13 we observe an improvement of approximately 8 dB in equivalent ISR relative to the Kelly detector when 5 epochs of the pre-whitening technique are applied with unit-modulus elements in $\boldsymbol{V}$, where the phases are chosen from a finite set.

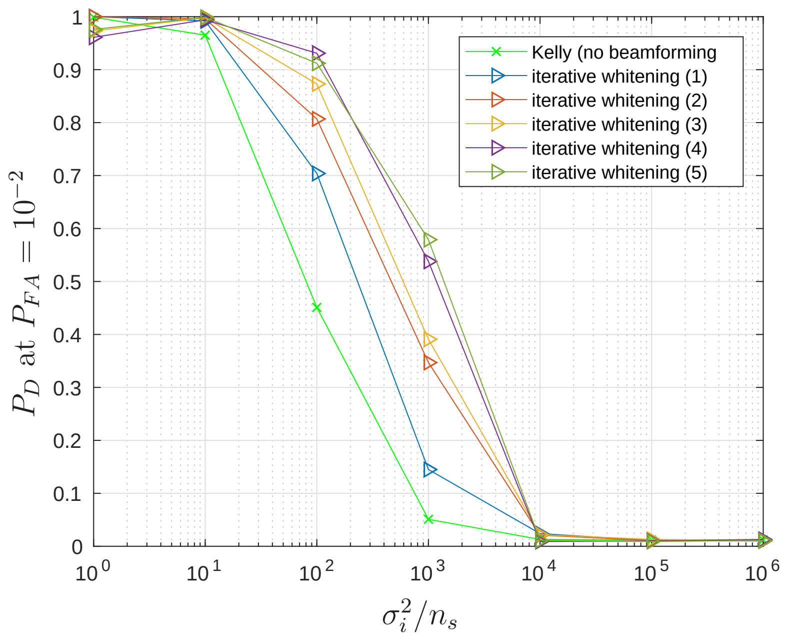


Figure 5.12: P_D vs σ_i^2/n_s with iterative whitening technique, where elements in $\boldsymbol{V}$ are unit-modulus.

5.6 Conclusion

In this chapter we considered the problem of adaptive detection from quantized measurements. We focused on the low-rank interference case, which models a small

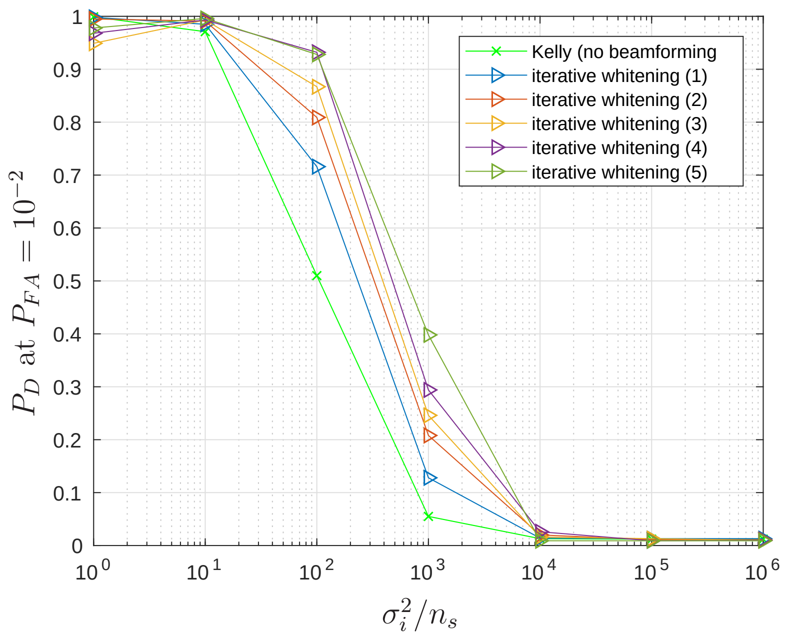


Figure 5.13: P_D vs σ_i^2/n_s with iterative whitening technique, where elements in $\boldsymbol{V}$ are unit-modulus and their phases are one of $4n_r$ uniformly spaced values.

number of jammers in a time-invariant channel. We first applied unquantized detection algorithms to this problem, and observed that in the low-interference regime, there was only a small loss in detection probability when using quantized measurements, but for any level of quantization, there is eventually an interference power large enough that will kill detection performance.

We then investigated detection performance when two conventional approaches to managing quantization were applied. In particular, we tested detection performance when a dither signal was added and when companding (i.e., non-uniform quantization) was used. In our experiments, dithering actually hurt detection performance, but companding improved detection performance. However, the improvement was small, and still did not prevent large interference from eventually killing detection performance.

Then, we developed the GLRT that assumed quantization. The GLRT for our quantized model involves solving two generalized bilinear inference problems. However, the overall detection performance of this GLRT was not good enough relative to simply applying unquantized detectors to quantized data to justify continuing its development.

In the final section of this chapter we applied various pre-processing techniques to the signal to effectively try to null the interference prior to quantization. We observed that methods that know the true interference plus noise covariance matrix are able to significantly improve detection performance, but they are not practical because in practice the true interference plus noise covariance matrix is unknown. In response, we developed an iterative technique for estimating the interference plus noise covariance matrix, and using this to apply a whitening filter prior to quantization. The whitening filter can then be iteratively improved over time as more measurements are taken. This approach offered the best improvement in detection performance in the high interference case relative to all of the techniques under evaluation.

Chapter 6: Conclusion

In this dissertation we considered various problems that involve inference in the generalized linear model. Specifically, we considered the multinomial logistic regression and the sketched clustering problems. In both applications, we applied the HyGAMP algorithm to estimate the underlying parameters of interest. Then, through numerical simulations on synthetic and real-world datasets, we demonstrated that our approximate message passing approach to inference outperformed existing state-of-the-art approaches based on optimization for inference in both accuracy and computational complexity.

Then, in the final chapter of this dissertation we considered the problem of adaptive detection from quantized measurements, while focusing on the low-rank interference case. We first studied the performance of unquantized detection approaches when they were given quantized measurements. We observed these unquantized techniques work well in the low interference case but not the high interference case. We then investigated various approaches to improving the detection performance in the high interference regime. We first considered approaches such has applying a dither signal or using non-uniform quantization (companding). Then, we derived and implemented the GLRT with the quantized model, which we gave rise to the generalized bilinear model. Another final method that we considered was to whiten the data prior to quantization using processing in the analog domain. Our particular approach used feedback from the output of the quantizer to design the analog whitening operation. Through numerical experiments, we observed this approach had the best improvement over among all approaches that were considered.

Bibliography

[1] D. L. Donoho, A. Maleki, and A. Montanari, “Message passing algorithms for compressed sensing,” *Proc. National Academy of Sciences*, vol. 106, no. 45, pp. 18 914–18 919, Nov. 2009.

[2] S. Rangan, “Generalized approximate message passing for estimation with random linear mixing,” in *Proc. IEEE Internat. Symposium on Information Theory*, Aug. 2011, pp. 2168–2172, (full version at *arXiv:1010.5141*).

[3] M. Bayati and A. Montanari, “The dynamics of message passing on dense graphs, with applications to compressed sensing,” *IEEE Trans. on Information Theory*, vol. 57, no. 2, pp. 764–785, Feb. 2011.

[4] J. P. Vila and P. Schniter, “Expectation-maximization Gaussian-mixture approximate message passing,” *IEEE Trans. on Signal Processing*, vol. 61, no. 19, pp. 4658–4672, Oct. 2013.

[5] S. Rangan, A. K. Fletcher, V. K. Goyal, E. Byrne, and P. Schniter, “Hybrid approximate message passing,” *IEEE Trans. on Signal Processing*, vol. 65, no. 17, pp. 4577–4592, 2017.

[6] J. T. Parker, P. Schniter, and V. Cevher, “Bilinear generalized approximate message passing—Part I: Derivation,” *IEEE Trans. on Signal Processing*, vol. 62, no. 22, pp. 5839–5853, Nov. 2014.

[7] T. Lesieur, F. Krzakala, and L. Zdeborová, “MMSE of probabilistic low-rank matrix estimation: Universality with respect to the output channel,” *arXiv:1507.03857*, Jan. 2016.

[8] T. Lesieur, F. Krzakala, and L. Zdeborov, “Constrained low-rank matrix estimation: Phase transitions, approximate message passing and applications,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2017, no. 7, p. 073403, 2017.

[9] E. M. Byrne and P. Schniter, “Sparse multinomial logistic regression via approximate message passing,” *IEEE Trans. on Signal Processing*, vol. 64, no. 21, pp. 5485–5498, 2016.

[10] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, pp. 957–968, Jun. 2005.

[11] N. Keriven, A. Bourrier, R. Gribonval, and P. Pérez, "Sketching for large-scale learning of mixture models," in *Proc. IEEE Internat. Conf. on Acoustics, Speech, and Signal Processing*, 2016, pp. 6190–6194.

[12] N. Keriven, N. Tremblay, Y. Traonmilin, and R. Gribonval, "Compressive K-means," *arXiv:1610.08738*, 2016.

[13] S. Rangan, A. K. Fletcher, V. K. Goyal, and P. Schniter, "Hybrid generalized approximate message passing with applications to structured sparsity," in *Proc. IEEE Internat. Symposium on Information Theory*, Jul. 2012, pp. 1236–1240, (full version at *arXiv:1111.2581*).

[14] E. M. Byrne, "Sparse multinomial logistic regression via approximate message passing," Master's thesis, The Ohio State University, Columbus, Ohio, July 2015.

[15] H. Sun *et al.*, "Neuronal and glioma-derived stem cell factor induces angiogenesis within the brain," *Cancer Cell*, vol. 9, pp. 287–300, 2006.

[16] A. Bhattacharjee *et al.*, "Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses," *Proc. National Academy of Sciences*, vol. 98, pp. 13 790–13 795, Nov 2001.

[17] J. Haxby, M. Gobbini, M. Furey, A. Ishai, J. Schouten, and P. Pietrini, "Distributed and overlapping representations of faces and objects in ventral temporal cortex," *Science*, vol. 293, pp. 2425–2430, 2001.

[18] K. A. Norman, S. M. Polyn, G. J. Detre, and J. V. Haxby, "Beyond mind-reading: multi-voxel pattern analysis of fMRI data," *Trends in Cognitive Sciences*, vol. 10, no. 9, pp. 424–430, Sep. 2006.

[19] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *J. Mach. Learn. Res.*, vol. 3, pp. 1289–1305, 2003.

[20] D. Lewis, Y. Yang, T. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, Apr. 2004.

[21] A. Gustafsson, A. Hermann, and F. Huber, *Conjoint Measurement: Methods and Applications*. Berlin: Springer-Verlag, 2007.

[22] Y. Plan and R. Vershynin, "Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach," *IEEE Trans. on Information Theory*, vol. 59, no. 1, pp. 482–494, 2013.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning.* New York: Springer, 2007.

[24] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[25] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale Bayesian logistic regression for text categorization," *Technometrics*, vol. 49, no. 3, pp. 291–304, Aug. 2007.

[26] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *J. Statst. Softw.*, vol. 33, no. 1, pp. 1–22, Jan. 2010.

[27] G. C. Cawley, N. L. C. Talbot, and M. Girolami, "Sparse multinomial logistic regression via Bayesian L1 regularisation," in *Proc. Neural Information Processing Systems Conf.*, 2007, pp. 209–216.

[28] L. Meier, S. van de Geer, and P. Bühlmann, "The group lasso for logistic regression," *Journal of the Royal Statistical Society: Series B*, vol. 70, pp. 53–71, 2008.

[29] Y. Grandvalet, "Least absolute shrinkage is equivalent to quadratic penalization," in *Proc. Internat. Conf. Artificial on Neural Networks*, 1998, pp. 201–206.

[30] D. J. C. MacKay, "The evidence framework applied to classification networks," *Neural Computation*, vol. 4, pp. 720–736, 1992.

[31] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* San Mateo, CA: Morgan Kaufman, 1988.

[32] J. Ziniel, P. Schniter, and P. Sederberg, "Binary classification and feature selection via generalized approximate message passing," *IEEE Trans. on Signal Processing*, vol. 63, no. 8, pp. 2020–2032, 2015.

[33] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. Motivation and construction," in *Proc. Information Theory Workshop*, Cairo, Egypt, Jan. 2010, pp. 1–5.

[34] D. A. Knowles and T. P. Minka, "Non-conjugate variational message passing for multinomial and binary regression," in *Proc. Neural Information Processing Systems Conf.*, 2011, pp. 1701–1709.

[35] A. Mousavi, A. Maleki, and R. G. Baraniuk, "Parameterless, optimal approximate message passing," *arXiv:1311.0035*, Nov. 2013.

[36] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artificial Intelligence*, vol. 42, pp. 393–405, 1990.

[37] A. Javanmard and A. Montanari, "State evolution for general approximate message passing algorithms, with applications to spatial coupling," *Information and Inference*, vol. 2, no. 2, pp. 115–144, 2013.

[38] S. Rangan, P. Schniter, E. Riegler, A. Fletcher, and V. Cevher, "Fixed points of generalized approximate message passing with arbitrary matrices," in *Proc. IEEE Internat. Symposium on Information Theory*, Jul. 2013, pp. 664–668, (full version at *arXiv:1301.6295*).

[39] S. Rangan, P. Schniter, and A. Fletcher, "On the convergence of generalized approximate message passing with arbitrary matrices," in *Proc. IEEE Internat. Symposium on Information Theory*, Jul. 2014, pp. 236–240, (preprint at *arXiv:1402.3210*).

[40] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B*, vol. 58, no. 1, pp. 267–288, 1996.

[41] D. R. Hunter and K. Lange, "A tutorial on MM algorithms," *The American Statistician*, vol. 58, no. 1, pp. 30–37, 2004.

[42] D. Hunter and R. Li, "Variable selection using MM algorithms," *Annals of Statistics*, vol. 33, no. 4, pp. 1617–1642, 2005.

[43] D. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.

[44] D. Böhning, "Multinomial logistic regression algorithm," *Ann. Inst. Statist. Math.*, vol. 44, pp. 197–200, 1992.

[45] P. Schniter, "Turbo reconstruction of structured sparse signals," in *Proc. Conf. on Information Science and Systems*, Princeton, NJ, Mar. 2010, pp. 1–6.

[46] L. A. Stefanski, "A normal scale mixture representation of the logistic distribution," *Stats. Prob. Letters*, vol. 11, no. 1, pp. 69–70, 1991.

[47] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* Cambridge, MA: MIT Press, 2006.

[48] C. M. Stein, "Estimation of the mean of a multivariate normal distribution," *Annals of Statistics*, vol. 9, pp. 1135–1151, 1981.

[49] M. I. Jordan, "Why the logistic function? A tutorial discussion on probabilities and neural networks," MIT, Computational Cognitive Science, Tech. Rep. 9503, 1995.

[50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, Nov. 1998, pp. 2278–2324.

[51] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay, "Clustering large graphs via the singular value decomposition," *Machine Learning*, vol. 56, no. 1-3, pp. 9–33, 2004.

[52] H. Steinhaus, "Sur la division des corps matériels en parties," *Bulletin de lAcadmie Polonaise des Sciences*, vol. 4, no. 12, pp. 801–804, 1956.

[53] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, Jun. 2010.

[54] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1027–1035.

[55] N. Keriven, A. Bourrier, R. Gribonval, and P. Pérez, "Sketching for large-scale learning of mixture models," *Information and Inference*, vol. 7, no. 3, pp. 447–508, 2017.

[56] N. Keriven, N. Tremblay, Y. Traonmilin, and R. Gribonval, "Compressive K-means," in *Proc. IEEE Internat. Conf. on Acoustics, Speech, and Signal Processing*, 2017, pp. 6369–6373.

[57] R. Gribonval, G. Blanchard, N. Keriven, and Y. Traonmilin, "Compressive statistical learning with random feature moments," *arXiv:1706.07180*, 2017.

[58] A. Feuerverger and R. A. Mureika, "The empirical characteristic function and its applications," *Annals of Statistics*, vol. 5, no. 1, pp. 88–97, 1977.

[59] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. Asilomar Conf. on Signals, Systems and Computers*, Pacific Grove, CA, 1993, pp. 40–44.

[60] P. McCullagh and J. A. Nelder, *Generalized Linear Models*, 2nd ed. London: Chapman & Hall/CRC, 1989.

[61] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd ed. New York: McGraw-Hill, 1991.

[62] M. Rudelson and R. Vershynin, "Hanson-Wright inequality and sub-Gaussian concentration," *Electron. Commun. Probab.*, vol. 18, no. 82, pp. 1–9, 2013.

[63] R. Gatto and S. R. Jammalamadaka, "The generalized von Mises distribution," *Statistical Methodology*, vol. 4, pp. 341–353, 2007.

[64] N. Keriven, N. Tremblay, and R. Gribonval, "SketchMLbox : a Matlab toolbox for large-scale learning of mixture models," 2016.

[65] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, pp. 83–97, 1955.

[66] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Neural Information Processing Systems Conf.*, 2001, pp. 849–856.

[67] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," in *Proc. ACM Intl. Conf. Multimedia*, 2010, pp. 1469–1472.

[68] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Intl. Conf. Comp. Vision Thy. Appl. (VISAPP)*, 2009, pp. 331–340.

[69] Y. Traonmilin, N. Keriven, R. Gribonval, and G. Blanchard, "Spikes super-resolution with random Fourier sampling," in *Proc. Workshop on Signal Processing with Adaptive Sparse Structured Representations (SPARS)*, 2017, pp. 1–2.

[70] E. Kelly, "An adaptive detection algorithm," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 22, no. 1, pp. 115–127, 1986.

[71] E. J. Kelly and K. M. Forsythe, "Adaptive detection and parameter estimation for multidimensional signal models," MIT Lincoln Laboratory, Lexington, MA, Tech. Rep. 848, Apr. 1989.

[72] K. Gerlach and M. J. Steiner, "Fast converging adaptive detection of Doppler-shifted, range-distributed targets," *IEEE Trans. on Signal Processing*, vol. 48, no. 9, pp. 2686–2690, 2000.

[73] B. Kang, V. Monga, and M. Rangaswamy, "Rank-constrained maximum likelihood estimation of structured covariance matrices," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 50, no. 1, pp. 501–515, 2014.

[74] L. T. McWhorter, "A high resolution detector in multi-path environments," in *Proc. Workshop on Adaptive Sensor Array Processing*, Lexington, MA, 2004.

[75] R. M. Gray and T. G. Stockham, Jr., "Dithered quantizers," *IEEE Trans. on Information Theory*, vol. 39, pp. 805–812, May 1993.

[76] M. Collins, S. Dasgupta, and R. E. Schapire, "A generalization of principal components analysis to the exponential family," in *Proc. Neural Information Processing Systems Conf.*, 2002, pp. 617–624.

[77] M. Udell, C. Horn, R. Zadeh, and S. Boyd, "Generalized low rank models," *Foundations and Trends in Machine Learning*, vol. 9, no. 1, pp. 1–118, 2016.

[78] J. de Leeuw, "Principal component analysis of binary data by iterated singular value decomposition," *Computational Statistics and Data Analysis*, vol. 50, no. 1, pp. 21–39, 2006.

[79] W. Feller, *An Introduction to Probability Theory and its Applications.* New York: Wiley, 1957.

[80] E. Demidenko, "Computational aspects of the probit model," *Mathematical Communications*, vol. 6, pp. 233–247, 2001.

[81] S. T. Smith, "Optimum phase-only adaptive nulling," *IEEE Trans. on Signal Processing*, vol. 47, no. 7, pp. 1835 – 1843, Jul. 1999.

[82] V. Venkateswaran and A. van der Veen, "Analog beamforming in MIMO communications with phase shift networks and online channel estimation," *IEEE Trans. on Signal Processing*, vol. 58, no. 8, Aug. 2010.