MESSAGE PASSING APPROACHES TO COMPRESSIVE INFERENCE UNDER STRUCTURED SIGNAL PRIORS

DISSERTATION

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

By

Justin Ziniel, B.S., M.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2014

Dissertation Committee:

Dr. Philip Schniter, Advisor

Dr. Lee C. Potter

Dr. Per Sederberg

ABSTRACT

Across numerous disciplines, the ability to generate high-dimensional datasets is driving an enormous demand for increasingly efficient ways of both capturing and processing this data. A promising recent trend for addressing these needs has developed from the recognition that, despite living in high-dimensional ambient spaces, many datasets have vastly smaller intrinsic dimensionality. When capturing (sampling) such datasets, exploiting this realization permits one to dramatically reduce the number of samples that must be acquired without losing the salient features of the data. When processing such datasets, the reduced intrinsic dimensionality can be leveraged to allow reliable inferences to be made in scenarios where it is infeasible to collect the amount of data that would be required for inference using classical techniques.

To date, most approaches for taking advantage of the low intrinsic dimensionality inherent in many datasets have focused on identifying succinct (i.e., sparse) representations of the data, seeking to represent the data using only a handful of "significant" elements from an appropriately chosen dictionary. While powerful in their own right, such approaches make no additional assumptions regarding possible relationships between the significant elements of the dictionary. In this dissertation, we examine ways of incorporating knowledge of such relationships into our sampling and processing schemes.

One setting in which it is possible to dramatically improve the efficiency of sampling schemes concerns the recovery of temporally correlated, sparse time series, and

in the first part of this dissertation we summarize our work on this important problem. Central to our approach is a Bayesian formulation of the recovery problem, which allows us to access richly expressive models of signal structure. While Bayesian sparse linear regression algorithms have often been shown to outperform their non-Bayesian counterparts, this frequently comes at the cost of substantially increased computational complexity. We demonstrate that, by leveraging recent advances in the field of probabilistic graphical models and message passing algorithms, we are able to dramatically reduce the computational complexity of Bayesian inference on structured sparse regression problems without sacrificing performance.

A complementary problem to that of efficient sampling entails making the most of the data that is available, particularly when such data is extremely scarce. Motivated by an application from the field of cognitive neuroscience, we consider the problem of binary classification in a setting where one has many possible predictive features, but only a small number of training examples. We build on the mathematical and software tools developed in the aforementioned regression setting, showing how these tools may be applied to classification problems. Specifically, we describe how inference on a generalized linear model can be conducted through our previously developed message passing framework, suitably modified to account for categorical response variables.

Dedicated to my parents

ACKNOWLEDGMENTS

This dissertation owes its existence to the contributions of many individuals, whom it is my great pleasure to acknowledge. First and foremost, I would like to thank my advisor, Philip Schniter, for providing an intellectually stimulating and challenging environment in which to work and play as I conducted this research. His curiosity, encouragement, engagement, work ethic, persistence, high standards, and patience heavily influenced the way in which I approach research questions, and I am tremendously grateful for the many years of guidance he provided me, and the highly accessible manner in which he provided it. I am also deeply indebted to Lee Potter for serving as my initial advisor when I began graduate school. His supportive, patient, warm, and friendly demeanor provided a very welcoming introduction into the world of graduate-level research, and his dedication to serving my best interests was evident and greatly appreciated.

I have been extremely fortunate to be a part of the Information Processing Systems Laboratory and enjoy the company of many intelligent, entertaining, and friendly labmates. Rohit Aggarwal, Evan Byrne, Brian Day, Brian Carroll, Ahmed Fasih, Amrita Ghosh, Derya Gol, Onur Gungor, Sung-Jun Hwang, Bin Li, Sugumar Murugesan, Wenzhuo Ouyang, Jason Palmer, Jason Parker, Naveen Ramakrishnan, Carl Rossler, Paul Schnapp, Mohammad Shahmohammadi, Subhojit Som, Arun Sridharan, Rahul Srivastava, Liza Toher, Jeremy Vila, and Yang (Tommy) Yang all provided many

hours of insightful, engaging, and oftentimes hilarious conversation and companionship. It was a sincere pleasure to interact with so many diverse people on a daily basis, and to be regularly enlightened, inspired, and gladdened by their presence.

In the course of conducting my research, I have been helped immensely by a number of individuals and institutions. I thank Jeri McMichael for (seemingly effortlessly) managing the bureaucratic aspects of being a graduate student at Ohio State, Per Sederberg for introducing me to the field of cognitive neuroscience and cheerfully serving on my Ph.D. committee, Sundeep Rangan for generously allowing myself and others to make use of, and contribute to, his `gampmatlab` software repository, Rizwan Ahmad for his help in learning the ins and outs of medical imaging, and countless colleagues around the world who took the time and energy to patiently answer my questions and make available their software and data. I also gratefully acknowledge the financial support that I recieved over the course of my graduate career from Ohio State, the Air Force Research Laboratory, MIT Lincoln Laboratory, and DARPA. The Ohio Supercomputer Center (OSC) also played a critical role in supporting the extensive computational needs of my research; many of the numerical experiments would have been infeasible without the tremendous resources provided by OSC.

My life during grad school has been enriched by a number of very close personal friends. I feel very lucky to have been able to spend a great deal of time with Brian Allen, Elbert Aull, Erin Bailey, Will Bates, Brandon Bell, Laura Black, Will Brintzenhoff, Patrick Collins, Matt Currie, Sarah Cusser, Jenai Cutcher, Lisa Dunn, Matt Ebright, Katy Greenwald, Brandon Groff, Greg Hostetler, Angela Kinnell, Matt Marlowe, Charles Mars, Jeff McKnight, Chris Page, Matt Parker, Adam Rusnak, Michelle Schackman, Jon Schick, María Sotomayor, Aliza Spaeth-Cook, Benjamin Valencia, Susan West, Joe Whittaker, Jun Yamaguchi, and Lara Yazvac. Aaron Greene is a master at reminding me to take a break from time to time to enjoy what

Columbus has to offer. I am grateful to Joe Pipia for his friendship, his pizza oven, and the many hours he has devoted over the years to helping me keep my house, my truck, and my backyard in good shape; I couldn't ask for a better neighbor. I've had many great times and good conversations with Eric Pitzer, and I am thankful that we have remained such close friends despite living on separate continents. Aimee Zisner has been a source of much happiness and unwavering support during the completion of this dissertation. I'm particularly grateful to Corey and Laura Aumiller for being a constant source of laughter, delicious meals, and much needed work breaks.

Lastly, I thank my family for providing me with the support and encouragement I needed to follow my dreams. Words can't express my gratitude.

VITA

2007	B.S. Electrical and Computer Engineering, The Ohio State University
2012	M.S. Electrical and Computer Engineering, The Ohio State University
2007 - 2014	Graduate Research Assistant, The Ohio State University

PUBLICATIONS

J. Ziniel, P. Sederberg, and P. Schniter, "Binary Linear Classification and Feature Selection via Generalized Approximate Message Passing," *Proc. Conf. on Information Sciences and Systems*, (Princeton, NJ), Mar. 2014. (Invited).

J. Ziniel and P. Schniter, "Dynamic Compressive Sensing of Time-Varying Signals via Approximate Message Passing," *IEEE Transactions on Signal Processing*, Vol. 61, No. 21, Nov. 2013.

J. Ziniel and P. Schniter, "Efficient High-Dimensional Inference in the Multiple Measurement Vector Problem," *IEEE Transactions on Signal Processing*, Vol. 61, No. 2, Jan. 2013.

J. Ziniel, S. Rangan, and P. Schniter, "A Generalized Framework for Learning and Recovery of Structured Sparse Signals," *IEEE Statistical Signal Processing Workshop*, (Ann Arbor, MI), Aug. 2012.

J. Ziniel and P. Schniter, "Efficient Message Passing-Based Inference in the Multiple Measurement Vector Problem," *Proc. Forty-fifth Asilomar Conference on Signals, Systems, and Computers (SS&C)*, (Pacific Grove, CA), Nov. 2011.

J. Ziniel, L. C. Potter, and P. Schniter, "Tracking and Smoothing of Time-Varying Sparse Signals via Approximate Belief Propagation," *Proc. Forty-fourth Asilomar Conference on Signals, Systems, and Computers (SS&C)*, (Pacific Grove, CA), Nov. 2010.

P. Schniter, L. C. Potter, and J. Ziniel, "Fast Bayesian Matching Pursuit: Model Uncertainty and Parameter Estimation for Sparse Linear Models," IPS Laboratory, The Ohio State University, Tech. Report No. TR-09-06 (Columbus, Ohio), Jun. 2009.

L. C. Potter, P. Schniter, and J. Ziniel, "A Fast Posterior Update for Sparse Underdetermined Linear Models," *Proc. Forty-second Asilomar Conference on Signals, Systems, and Computers (SS&C)*, (Pacific Grove, CA), Oct. 2008.

L. C. Potter, P. Schniter, and J. Ziniel, "Sparse Reconstruction for Radar," *Algorithms for Synthetic Aperture Radar Imagery XV, Proc. SPIE*, E. G. Zelnio and F. D. Garber, Eds., vol. 6970, 2008..

P. Schniter, L. C. Potter, and J. Ziniel, "Fast Bayesian Matching Pursuit," *Proc. Workshop on Information Theory and Applications (ITA)*, (La Jolla, CA), Jan. 2008.

FIELDS OF STUDY

Major Field: Electrical and Computer Engineering

Specializations: Digital Signal Processing, Machine Learning

TABLE OF CONTENTS

Abstract ii

Dedication iii

Acknowledgments v

Vita viii

List of Figures xiii

List of Tables xv

CHAPTER PAGE

1 Introduction 1

1.1 Compressed Sensing 2
1.2 Bayesian Inference 5
1.3 Our Contributions 8
1.3.1 The Multiple Measurement Vector Problem 9
1.3.2 The Dynamic Compressed Sensing Problem 11
1.3.3 Binary Classification and Structured Feature Selection . . 12

2 The Multiple Measurement Vector Problem 13

2.1 Introduction 13
2.1.1 Notation 17
2.2 Signal Model 18
2.3 The Support-Aware Kalman Smoother 20
2.4 The AMP-MMV Algorithm 21
2.4.1 Message Scheduling 23
2.4.2 Implementing the Message Passes 25
2.5 Estimating the Model Parameters 34
2.6 Numerical Study 37
2.6.1 Performance Versus Sparsity, M/K 39
2.6.2 Performance Versus T 41

2.6.3 Performance Versus SNR . . . 42
2.6.4 Performance Versus Undersampling Rate, N/M . . . 43
2.6.5 Performance Versus Signal Dimension, N . . . 45
2.6.6 Performance With Time-Varying Measurement Matrices . . . 46
2.7 Conclusion . . . 47

3 The Dynamic Compressive Sensing Problem . . . 49

3.1 Introduction . . . 49
3.2 Signal Model . . . 53
3.3 The DCS-AMP Algorithm . . . 56
3.3.1 Message scheduling . . . 59
3.3.2 Implementing the message passes . . . 60
3.4 Learning the signal model parameters . . . 63
3.5 Incorporating Additional Structure . . . 64
3.6 Numerical Study . . . 68
3.6.1 Performance across the sparsity-undersampling plane . . . 70
3.6.2 Performance vs p_{01} and α . . . 72
3.6.3 Recovery of an MRI image sequence . . . 74
3.6.4 Recovery of a CS audio sequence . . . 77
3.6.5 Frequency Estimation . . . 79
3.7 Conclusion . . . 81

4 Binary Classification, Feature Selection, and Message Passing . . . 83

4.1 Introduction . . . 83
4.1.1 Notation . . . 87
4.2 Generalized Approximate Message Passing . . . 87
4.2.1 Sum-Product GAMP . . . 87
4.2.2 Max-Sum GAMP . . . 90
4.2.3 GAMP Summary . . . 93
4.3 Predicting Misclassification Rate via State Evolution . . . 93
4.4 GAMP for Classification . . . 96
4.4.1 Logistic Classification Model . . . 96
4.4.2 Probit Classification Model . . . 98
4.4.3 Hinge Loss Classification Model . . . 99
4.4.4 A Method to Robustify Activation Functions . . . 100
4.4.5 Weight Vector Priors . . . 100
4.4.6 The GAMP Software Suite . . . 103
4.5 Automatic Parameter Tuning . . . 103
4.5.1 Traditional EM Parameter Tuning . . . 104
4.5.2 Tuning via Bethe Free Entropy Minimization . . . 106
4.6 Numerical Study . . . 109
4.6.1 Text Classification and Adaptive Learning . . . 109

4.6.2 Robust Classification . . . 111
4.6.3 Multi-Voxel Pattern Analysis . . . 113
4.7 Conclusion . . . 116

Bibliography . . . 118

APPENDICES

A The Basics of Belief Propagation and (G)AMP . . . 129

B Taylor Series Approximation of $\nu^{\text{mod}}_{f_n^{(t)} \to \theta_n^{(t)}}$ . . . 133

C DCS-AMP Message Derivations . . . 135

C.1 Derivation of **(into)** Messages . . . 136
C.2 Derivation of **(within)** Messages . . . 138
C.3 Derivation of Signal MMSE Estimates . . . 141
C.4 Derivation of AMP update equations . . . 142
C.5 Derivation of **(out)** Messages . . . 143
C.6 Derivation of Forward-Propagating **(across)** Messages . . . 146
C.7 Derivation of Backward-Propagating **(across)** Messages . . . 147

D DCS-AMP EM Update Derivations . . . 149

D.1 Sparsity Rate Update: λ^{k+1} . . . 150
D.2 Markov Transition Probability Update: p_{01}^{k+1} . . . 151
D.3 Amplitude Mean Update: ζ^{k+1} . . . 152
D.4 Amplitude Correlation Update: α^{k+1} . . . 153
D.5 Perturbation Variance Update: ρ^{k+1} . . . 154
D.6 Noise Variance Update: $(\sigma_e^2)^{k+1}$ . . . 155

E GAMP Classification Derivations . . . 156

E.1 Derivation of State Evolution Output Covariance . . . 156
E.2 Sum-Product GAMP Updates for a Logistic Likelihood . . . 158
E.3 Sum-Product GAMP Updates for a Hinge Likelihood . . . 161
E.4 Sum-Product GAMP Updates for a Robust-p^* Likelihood . . . 166
E.5 EM Learning of Robust-p^* Label Corruption Probability . . . 167
E.6 EM Update of Logistic Scale, α . . . 169
E.7 Bethe Free Entropy-based Update of Logistic Scale, α . . . 171
E.8 EM Update of Probit Variance, v^2 . . . 172

F Miscellaneous GAMP/turboGAMP Derivations . . . 173

F.1 EM Learning of Rate Parameter for a Laplacian Prior . . . 173
F.2 Sum-Product GAMP Equations for an Elastic Net Prior . . . 176

LIST OF FIGURES

FIGURE		PAGE
2.1	Factor graph representation of AMP-MMV signal model	22
2.2	A summary of the four AMP-MMV message passing phases	26
2.3	AMP-MMV performance versus normalized sparsity rate	40
2.4	AMP-MMV performance versus normalized sparsity rate, low α	41
2.5	AMP-MMV performance versus number of timesteps	42
2.6	AMP-MMV performance versus SNR	43
2.7	AMP-MMV performance versus undersampling rate	44
2.8	AMP-MMV performance versus signal dimension	46
2.9	AMP-MMV performance versus changing measurement matrices	48
3.1	Factor graph representation of DCS-AMP signal model	57
3.2	A summary of the four DCS-AMP message passing phases	61
3.3	DCS-AMP performance across sparsity-undersampling plane	72
3.4	DCS-AMP performance versus signal dynamicity	73
3.5	Dynamic MRI image sequence, and recoveries	76
3.6	DCT coefficient magnitudes (in dB) of an audio signal.	78
4.1	Factor graph representation of a classification problem	89
4.2	State-evolution prediction of GAMP classification error rate	95
4.3	Robust classification performance, $M \gg N$	113
A.1	The factor graph representation of the decomposition of (A.2).	130

A.2 The GAMP system model. 131
C.1 A summary of the four DCS-AMP message passing phases 136
E.1 Factor graph for Robust-p^* hidden variable 168

LIST OF TABLES

TABLE PAGE

2.1 A legend for the AMP-MMV factor graph 23

2.2 Pseudocode for the AMP-MMV algorithm 30

2.3 Pseudocode function for computing Taylor approximation of (2.12) . 33

2.4 EM algorithm update equations for the AMP-MMV signal model . . 36

3.1 A legend for the DCS-AMP factor graph 57

3.2 Pseudocode for the DCS-AMP algorithm 64

3.3 EM algorithm update equations for the DCS-AMP signal model . . . 65

3.4 Performance on MRI dataset, increased initial sampling 76

3.5 Performance on MRI dataset, uniform subsampling 77

3.6 Performance on audio CS dataset 79

3.7 Performance on synthetic frequency estimation task 82

4.1 Sum-product GAMP computations for probit activation function. . . 98

4.2 Sum-product GAMP computations for a robust activation function . 101

4.3 GAMP computations for the elastic-net regularizer 102

4.4 Definitions of elastic-net quantities 102

4.5 GAMP classification likelihood modules 103

4.6 GAMP classification weight vector prior modules 103

4.7 Classification performance on a text classification problem 111

4.8 Classification performance on an fMRI classification problem 115

D.1 Distributions underlying the probabilistic model of DCS-AMP 150

CHAPTER 1

INTRODUCTION

"Data! Data! Data! I can't make bricks without clay."

- Sherlock Holmes

In 2011, an IDC Digital Universe study estimated that over 1.8 trillion gigabytes (GB) of digital content would be generated over the course of the year [1]. This volume of raw data, if stored on 32GB Apple iPads, would require 57.5 billion such devices, enough to create a wall 4,005 miles long and 61 feet high, extending from Anchorage, Alaska to Miami, Florida. Without question, society has entered the era of "Big Data". The dramatic growth in the size and scope of large datasets is the result of many factors, including the proliferation of sensors and data acquisition devices, declining storage costs, and the growth of web and cloud infrastructure.

In order to properly take advantage of the tremendous volume of data, a number of important challenges must be addressed. Two critical, and highly interrelated, challenges concern the acquisition and interpretation of such data. In a number of Big Data applications, cost is a fundamental constraint in the acquisition process. Consider the following examples: Many camera systems are capable of generating gigabytes of raw data in a short period of time—data which is then immediately compressed in order to discard unnecessary and redundant information. In some applications, this sample-then-compress paradigm is acceptable, while in others, where hardware costs dominate (e.g., hyperspectral imaging), it is much more preferable to

compress *while* sampling, that is, collect only the truly necessary information. Similarly, in cardiac MRI, long acquisition times limit spatial and temporal resolution, motivating a desire for the ability to reduce the number of measurements that must be acquired without sacrificing image quality. As a final example, large corpora of text classification datasets are expensive to generate, since each training example must be hand-labeled—a time-consuming process.

All of the aforementioned examples possess several unifying traits. First, each problem entails working with very high-dimensional data. Images containing many millions of pixels are routine, and text classification problems typically involve hundreds of thousands, or even millions, of attributes associated with each training document. Second, there is a strong motivation to reduce the amount of information that must be measured, preferably without sacrificing the ability to make useful inferences from the data. Third, each of these examples incorporates an *inverse problem* into the process of interpreting the data.

1.1 Compressed Sensing

A particular inverse problem of interest in this dissertation is that of *compressed sensing* (CS) [2–4]. At its most elemental, CS represents an alternative to the aforementioned sample-then-compress data collection paradigm, namely, sample-*while*-compressing. Traditional sampling theory conforms more to the former paradigm. In particular, Shannon-Nyquist theory informs us that samples acquired uniformly in time or space at twice the highest signal bandwidth (or desired resolution) can be used to accurately reconstruct the signal through a simple, computationally inexpensive process known as sinc interpolation. CS differs from Shannon-Nyquist sampling in several important respects. First, it is primarily studied as a finite-dimensional, digital-to-digital sampling scheme, although continuous-time sampling

is possible within the theory [4]. Second, CS requires a more complex sampling process; rather than acquiring point samples uniformly in time/space, CS collects samples in the form of inner products between the complete signal and a series of "test waveforms." Third, unlike sinc interpolation, the CS inverse problem is highly non-linear in nature, requiring more complex reconstruction algorithms. In exchange for these tradeoffs, CS offers the ability to dramatically reduce the number of samples that must be acquired without sacrificing reconstruction fidelity.

Mathematically, if $\boldsymbol{y} \in \mathbb{C}^M$ denotes a length-M complex-valued observation vector, and $\boldsymbol{x} \in \mathbb{C}^N$ denotes a length-N signal, then a generic (linear) sampling process can be represented as

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

where $\boldsymbol{A} \in \mathbb{C}^{M \times N}$ represents a *known* linear transformation of the signal, and $\boldsymbol{e}$ represents (unknown) corrupting noise. When $M \geq N$, corresponding to traditional sampling, (1.1) represents an overdetermined linear system of equations, for which a number of tools exist for estimating $\boldsymbol{x}$ from $\boldsymbol{y}$. When $M < N$, the primary setting of interest in CS, additional information about $\boldsymbol{x}$ must be brought to bear in order to make the inverse problem well-posed.

The fundamental insight offered by CS is that, while Shannon-Nyquist sampling enables the recovery of *any* bandlimited signal, most real-world signals possess additional structure beyond being bandlimited. In particular, the "intrinsic" dimensionality of such signals is vastly below their ambient dimensionality. That most high-dimensional signals are compressed after acquisition is a testament to this fact. The principle of *sparsity* is therefore central to CS.

Sparsity reflects the notion that, in a suitably chosen basis, $\boldsymbol{\Psi} \in \mathbb{C}^{N \times N}$, the underlying signal being recovered has a succinct representation, i.e., there exists a

K-sparse vector $\boldsymbol{\alpha} \in \mathbb{C}^N$, $\|\boldsymbol{\alpha}\|_0 = K \ll N$, such that

$$\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{\alpha}. \tag{1.2}$$

Equation (1.2) expresses the idea that the signal $\boldsymbol{x}$ is composed of only a handful of columns (or "atoms") of $\boldsymbol{\Psi}$ corresponding to the non-zero entries of $\boldsymbol{\alpha}$. Without loss of generality, in this dissertation we will generally assume $\boldsymbol{x}$ is sparse in the canonical (identity) basis, and thus $\|\boldsymbol{x}\|_0 \ll N$. Revisiting (1.1), any sparsifying basis $\boldsymbol{\Psi}$ can be incorporated into $\boldsymbol{A}$ via $\boldsymbol{A} = \boldsymbol{\Phi}\boldsymbol{\Psi}$.

In the early days of CS, the primary inference objective was to estimate $\boldsymbol{x}$ by seeking the sparsest representation consistent with the observations, e.g.,

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{x}\|_0 \quad \text{s.t.} \quad \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 < \varepsilon, \tag{1.3}$$

where ε is a parameter that controls the fidelity of the recovery. Unfortunately, solving (1.3) is an NP-hard optimization problem, motivating researchers to consider alternative approaches that would achieve the same end. The literature on such approaches is quite vast (see [5] for a partial summary), but a general taxonomy can be deduced: convex optimization approaches, which solve relaxed versions of (1.3) (e.g., the Lasso [6]); greedy algorithms, which iteratively add elements to the signal support set until a convergence criterion is met; and probabilistic algorithms, which frame the recovery problem as a signal estimation problem, with a signal prior distribution chosen to encourage sparse solutions.

While the early CS algorithms broke important theoretical and application-driven ground, in more recent years it has become increasingly evident that additional performance gains (in the form of further reductions in sampling rates, or increases in

signal-to-noise ratio (SNR)) can be obtained by exploiting additional signal structure that goes beyond "simple sparsity".[1] This additional structure is present in a wide range of applications, including magnetoencephalography (MEG) [7], dynamic MRI [8], and underwater communications [9], and manifests itself primarily through complex relationships between signal coefficients (e.g., the phenomenon of "persistence across scale" observed in wavelet decompositions of natural images [10]). Duarte and Eldar [11] provide a comprehensive review of such contemporary "structured CS" research thrusts.

1.2 Bayesian Inference

While any technique will have advantages and disadvantages in different applications, we argue that Bayesian approaches represent the most promising avenue for the development of future structured sparse inference algorithms. A primary motivator for this argument is the substantial body of empirical evidence (e.g., [12–16]) that has demonstrated that Bayesian approaches offer superior performance (w.r.t. mean square error (MSE)) over non-Bayesian techniques. Although the success of a Bayesian method will depend in no small part on how well its signal model is matched to the true signal, non-Bayesian approaches are likewise sensitive to the statistics of the true signal, and in certain instances can be viewed as implicitly seeking maximum a posteriori (MAP) estimates under certain families of priors. Indeed, one might consider Bayesian methods to be more robust to model mismatch than their non-Bayesian counterparts, since Bayesian methods view departures from the assumed model of signal structure as unlikely, but not impossible, events, whereas

[1] By "simple sparsity," we mean signals for which all support sets of a given cardinality are equiprobable.

non-Bayesian structured sparsity models, e.g., union-of-subspaces [11], can enforce a rigid "all or nothing" conformity to the assumed type of signal structure.

A closely related advantage of the Bayesian paradigm in structured CS problems concerns the task of characterizing the structure inherent in the underlying signal. By adopting a Bayesian mindset, we provide ourselves access to the richly expressive framework of *probabilistic graphical models* (PGMs). The PGM framework is advantageous for describing complex forms of signal structure because, as noted by Koller and Friedman [17, §1.2.2], "the type of representation provided by this framework is transparent, in that a human expert can understand and evaluate its semantics and properties. This property is important for constructing models that provide an accurate reflection of our understanding of a domain. Models that are opaque can easily give rise to unexplained, and even undesirable, answers." PGMs offer us a way to translate qualitative descriptions of structure into concrete mathematical language while still remaining flexible enough to accommodate deviations from our model.

One important characteristic of all structured sparsity algorithms is the dependence of performance on model/algorithm parameters. Algorithms that fall within the Bayesian framework rely on a statistical model of structure that is characterized by a collection of parameters and hyperparameters. By nature of their Bayesian formulation, a number of approaches are available to automatically learn model parameters from the data through, e.g., an expectation-maximization algorithm [13, 14, 18, 19], or margin them out completely [20, 21]. In contrast, while non-Bayesian methods retain a dependence on algorithmic parameters, little attention has been devoted to finding ways to set these parameters in a principled manner. Consequently, one is left with a dilemma when applying these non-Bayesian techniques: either fix the parameters according to some heuristic criteria and hope that they are near-optimal,

or else resort to a computationally expensive cross-validation procedure. The ability to automatically tune parameters is thus another strong advantage of Bayesian algorithms.

Given the number of Bayesian approaches to solving CS problems, one might assume that there is little room for improvement. On the contrary, our research to-date has sought to alleviate one of the primary deficiencies of Bayesian methods: their high computational complexity relative to non-Bayesian schemes.[2]

As a very coarse rule, in most structured sparse regression problems, the computational complexity of different methods can be rank-ordered (from fastest to slowest) as follows:[3]

1. *Greedy methods* (e.g., OMP: $\mathcal{O}(MNK)$ [22])
2. *Convex relaxations* (e.g., Lasso: $\mathcal{O}(M^2N)$ [6]; IHT: $\mathcal{O}(MN + K^2M)$ [22])
3. *Bayesian methods* (e.g., approaches involving covariance matrix inversion: $\mathcal{O}(N^2)$; BCS: $\mathcal{O}(M^2N)$ [23])

The relative performance of these methods appears to be ordered inversely, reflecting the fact that "there's no free lunch".

The reasons why Bayesian methods are slower than their non-Bayesian counterparts on structured sparse inference problems are varied. In the case of approaches that employ empirical Bayesian strategies, such as the Sparse Bayesian Learning paradigm [24], the computational costs typically arise from the need to invert covariance matrices in Gaussian models. In Markov chain Monte Carlo (MCMC) approaches to approximating a full posterior distribution using a collection of samples

[2]In comparing computational complexity, we are ignoring the potentially expensive parameter tuning processes of non-Bayesian algorithms, since these costs will vary depending on the specific algorithm and application at hand.

[3]For illustrative purposes, we are reporting complexity for common unstructured/traditional CS algorithms, where M is the number of measurements, N is the number of unknowns, and K is the number of non-zero signal coefficients.

drawn from the distribution, complexity may result from matrix inversion, or from the slow convergence that can be encountered in high-dimensional inference settings. It is precisely this challenge of devising computationally efficient and highly accurate Bayesian inference methods for high-dimensional CS problems that we have been addressing through our work.

1.3 Our Contributions

In what follows, we highlight three structured CS research problems that comprise the focus of this dissertation. The first two problems align with the traditional CS focus of sparse linear regression, while the final problem moves CS from the domain of regression to that of classification. Several common themes can be found across our work on these problems. First, a great deal of emphasis is placed on designing algorithms that are computationally tractable for high-dimensional inference problems. This emphasis on tractability motivates the application of recently developed *approximate message passing* (AMP) [25,26] and *generalized AMP* (GAMP) [27] techniques, which leverage concentration of measure phenomena in high-dimensions in order to approximate otherwise intractable loopy belief propagation (loopy BP) [28] methods. The resultant first-order algorithms are fast, highly accurate, and admit to rigorous analysis (see, e.g., [29, 30]). A comprehensive discussion of loopy BP and (G)AMP is beyond the scope of this dissertation, however, the interested reader can find a brief primer in Appendix A.

A second common theme across our work is the adoption of a Bayesian inference strategy. As noted in Section 1.2, there are a number of compelling reasons to consider a Bayesian approach, including the ability to adopt a PGM framework. Our use of PGMs to model the various forms of structure is intimately connected to our chosen methods of inference; the close ties between (G)AMP and belief propagation

are exploited in our algorithms by embedding (G)AMP inference within larger PGMs that model the relevant statistical structure. This provides us with a great deal of flexibility and modularity in three aspects: how we model structure, how we conduct inference, and how we implement our algorithms in software.

A third and final theme is our use of automatic parameter tuning techniques. In Section 1.2, we identified Bayesian systems' automatic parameter tuning capabilities as a desirable alternative to other tuning schemes such as cross-validation. Therefore, in developing our algorithms, we went to great lengths to ensure that we could not only specify structured statistical models, but also learn the parameters of those models adaptively from the data. Fortunately, the PGM framework undergirding our algorithms readily lends itself to such automatic tuning, particularly via expectation-maximization techniques. We find that, in most cases, the quantities that must be computed in order to update model parameter estimates are available as a byproduct of our message passing-based primary inference process. This ability to tune parameters in a principled manner from the data often sets our methods apart from competing approaches, offering the end-user an alternative to expensive and time-consuming cross-validation or heuristic hand-tuning.

1.3.1 The Multiple Measurement Vector Problem

In Chapter 2, we explore one particular structured CS problem that is a generalization of the single measurement vector (SMV) CS problem (1.1), known as the *multiple measurement vector* (MMV) problem [31, 32]. As the name implies, in the MMV CS problem, one seeks to recover a collection of sparse signals, $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(T)}$ from another collection of undersampled linear measurements, $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(T)}$. The crux of the MMV problem is in the additional structure that is imposed on the sparse collection of signal vectors $\{\boldsymbol{x}^{(t)}\}_{t=1}^{T}$.

In its earliest incarnation [7], the MMV problem was viewed as a problem of recovering a row-sparse signal matrix $\boldsymbol{X} = [\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}]$ from observations $\boldsymbol{Y} = [\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(T)}]$ obtained via

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{E}, \tag{1.4}$$

when all signal vectors $\boldsymbol{x}^{(t)}$ are assumed to be "jointly sparse", i.e., $\text{supp}(\boldsymbol{x}^{(t)}) = \text{supp}(\boldsymbol{x}^{(s)}) \, \forall \, t, \, s$. Such a problem finds application in magnetoencephalography [7,12], direction-of-arrival estimation [33], and parallel MRI [34]. The benefits afforded by modeling joint sparsity can be significant; under mild conditions, the probability of failing to accurately recover $\boldsymbol{X}$ decays exponentially in the number of measurement vectors, T, when accounting for the joint sparsity [35].

Our proposed algorithm, AMP-MMV, leverages AMP algorithmic techniques to enable accurate, rapid recovery of the unknown $\boldsymbol{X}$. Key features of our approach, as compared to alternative MMV solvers, include:

1. A statistical model of the MMV problem that explicitly incorporates amplitude correlation within the non-zero rows of $\boldsymbol{X}$.
2. A computational complexity that grows only linearly in all problem dimensions, yielding unrivaled speed amongst state-of-the-art Bayesian algorithms.
3. A principled scheme for automatically tuning the parameters of our statistical model, based on the available data.
4. An ability to support heterogeneous measurement matrices, (i.e., $\boldsymbol{A}^{(t)}$ instead of $\boldsymbol{A}$).
5. Performance on-par with that of an oracle-aided Kalman smoother that lower-bounds the achievable MSE.

1.3.2 The Dynamic Compressed Sensing Problem

The jointly sparse MMV problem can be generalized to allow for small changes in the support over time, as well as time-varying measurement matrices, e.g.,

$$\boldsymbol{y}^{(t)} = \boldsymbol{A}^{(t)}\boldsymbol{x}^{(t)} + \boldsymbol{e}^{(t)}, \qquad t = 1, \ldots, T \tag{1.5}$$

This generalized MMV problem, known as the *dynamic CS* problem [36,37], is prevalent in applications in which the sparse signal vectors $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots$ describe a slowly time-varying phenomenon, such as dynamic MRI [8].

In Chapter 3, we describe a novel, computationally efficient Bayesian algorithm designed to solve the dynamic CS problem. As with AMP-MMV, our proposed dynamic CS algorithm, DCS-AMP, employs an AMP algorithm as a sub-routine to perform the computationally intensive inference tasks. Unique advantages of DCS-AMP include:

1. The ability to perform soft (i.e., probabilistic) signal estimation and support detection, which alleviates the problem of errors accumulating over time due to erroneous hard estimates.
2. A computational complexity that grows only linearly in all problem dimensions, yielding unrivaled speed amongst state-of-the-art Bayesian algorithms.
3. A common framework for performing both causal filtering and non-causal smoothing of sparse time-series.
4. A principled scheme for automatically tuning the parameters of our statistical model, based on the available data.
5. Strong algorithmic connections to an oracle-aided Kalman smoother that lower-bounds the achievable MSE.

1.3.3 Binary Classification and Structured Feature Selection

In Chapters 2 and 3, we considered two structured sparse linear regression problems. In Chapter 4, we turn our attention to the complementary problem of binary linear classification and structured feature selection. In *binary linear classification*, our objective is to learn a hyperplane, defined by the normal vector $\boldsymbol{w}$, that separates $\mathbb{R}^N$ into two half-spaces, for the purpose of predicting a discrete class label $y \in \{-1, 1\}$ associated with a vector of quantifiable features $\boldsymbol{x} \in \mathbb{R}^N$ from a mapping $g(z) : \mathbb{R} \rightarrow \{-1, 1\}$ of the linear "score" $z \triangleq \langle \boldsymbol{x}, \boldsymbol{w} \rangle$. The goal of *structured feature selection* is to identify a subset of the N feature weights in $\boldsymbol{w}$ that contain the bulk of the discriminatory power for segregating the two classes. In particular, the identified subset is expected to possess non-trivial forms of structure, e.g., a spatial clustering of discriminative features in an image classification task.

While the mapping from z to y in Chapters 2 and 3 consisted of the addition of additive white Gaussian noise (AWGN), in binary linear classification the mapping is complicated by the discrete nature of the output y. The original AMP [25, 26] framework was intended to work only in the AWGN setting. Fortunately, GAMP [27] extends AMP to non-AWGN, generalized-linear mappings, such as $g(z)$, allowing us to leverage the design principles developed in previous chapters. Our work represents the first study of GAMP's suitability for classification tasks, and our contributions include:

1. Implementation of several popular binary classifiers, including logistic and probit classifiers, and support vector machines.
2. A characterization of GAMP's misclassification rate under various weight vector priors, $p(\boldsymbol{w})$, using GAMP's state evolution formalism.
3. A principled scheme for automatically tuning model parameters that govern the bias-variance tradeoff, based on the available training data.

CHAPTER 2

THE MULTIPLE MEASUREMENT VECTOR PROBLEM

"It is of the highest importance in the art of detection to be able to recognize, out of a number of facts, which are incidental and which vital. Otherwise your energy and attention must be dissipated instead of being concentrated."

- Sherlock Holmes

In this chapter,[1] we develop a novel Bayesian algorithm designed to solve the *multiple measurement vector* (MMV) problem, which generalizes the sparse linear regression, or *single measurement vector* (SMV), problem to the case where a group of measurement vectors has been obtained from a group of signal vectors that are assumed to be jointly sparse—sharing a common support. Such a problem has many applications, including magnetoencephalography [7, 12], direction-of-arrival estimation [33] and parallel magnetic resonance imaging (pMRI) [34].

2.1 Introduction

Mathematically, given T length-M measurement vectors, the traditional MMV objective is to recover a collection of length-N sparse vectors $\{\boldsymbol{x}^{(t)}\}_{t=1}^{T}$, when $M < N$.

[1] Work presented in this chapter is largely excerpted from a journal publication co-authored with Philip Schniter, entitled "Efficient High-Dimensional Inference in the Multiple Measurement Vector Problem." [32]

Each measurement vector, $\boldsymbol{y}^{(t)}$, is obtained as

$$\boldsymbol{y}^{(t)} = \boldsymbol{A}\boldsymbol{x}^{(t)} + \boldsymbol{e}^{(t)}, \quad t = 1, \ldots, T, \tag{2.1}$$

where $\boldsymbol{A}$ is a known measurement matrix and $\boldsymbol{e}^{(t)}$ is corrupting additive noise. The unique feature of the MMV problem is the assumption of joint sparsity: the support of each sparse signal vector $\boldsymbol{x}^{(t)}$ is identical. Oftentimes, the collection of measurement vectors forms a time-series, thus we adopt a temporal viewpoint of the MMV problem, without loss of generality.

A straightforward approach to solving the MMV problem is to break it apart into independent SMV problems and apply one of the many SMV algorithms. While simple, this approach ignores valuable temporal structure in the signal that can be exploited to provide improved recovery performance. Indeed, under mild conditions, the probability of recovery failure can be made to decay exponentially as the number of timesteps T grows, when taking into account the joint sparsity [35].

Another approach (e.g., [38]) to the joint-sparse MMV problem is to restate (2.1) as the block-sparse SMV model

$$\bar{\boldsymbol{y}} = \mathcal{D}(\boldsymbol{A})\bar{\boldsymbol{x}} + \bar{e}, \tag{2.2}$$

where $\bar{\boldsymbol{y}} \triangleq \left[\boldsymbol{y}^{(1)^\mathsf{T}}, \ldots, \boldsymbol{y}^{(T)^\mathsf{T}}\right]^\mathsf{T}$, $\bar{\boldsymbol{x}} \triangleq \left[\boldsymbol{x}^{(1)^\mathsf{T}}, \ldots, \boldsymbol{x}^{(T)^\mathsf{T}}\right]^\mathsf{T}$, $\bar{\boldsymbol{e}} \triangleq \left[\boldsymbol{e}^{(1)^\mathsf{T}}, \ldots, \boldsymbol{e}^{(T)^\mathsf{T}}\right]^\mathsf{T}$, and $\mathcal{D}(\boldsymbol{A})$ denotes a block diagonal matrix consisting of T replicates of $\boldsymbol{A}$. In this case, $\boldsymbol{x}$ is block-sparse, where the n^{th} block (for $n = 1, \ldots, N$) consists of the coefficients $\{x_n, x_{n+N}, \ldots, x_{n+(T-1)N}\}$. Equivalently, one could express (2.1) using the matrix model

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{E}, \tag{2.3}$$

where $\boldsymbol{Y} \triangleq \left[\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(T)}\right]$, $\boldsymbol{X} \triangleq \left[\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}\right]$, and $\boldsymbol{E} \triangleq \left[\boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(T)}\right]$. Under the matrix model, joint sparsity in (2.1) manifests as row-sparsity in $\boldsymbol{X}$. Algorithms developed for the matrix MMV problem are oftentimes intuitive extensions of SMV algorithms, and therefore share a similar taxonomy. Among the different techniques that have been proposed are mixed-norm minimization methods [12, 39–41], greedy pursuit methods [12, 42, 43], and Bayesian methods [13, 14, 21, 33, 44]. Existing literature suggests that greedy pursuit techniques are outperformed by mixed-norm minimization approaches, which in turn are surpassed by Bayesian methods [12–14].

In addition to work on the MMV problem, related work has been performed on a similar problem sometimes referred to as the "*dynamic CS*" problem [36, 45–48]. The dynamic CS problem also shares the trait of working with multiple measurement vectors, but instead of joint sparsity, considers a situation in which the support of the signal changes slowly over time.

Given the plethora of available techniques for solving the MMV problem, it is natural to wonder what, if any, improvements can be made. In this chapter, we will primarily address two deficiencies evident in the available MMV literature. The first deficiency is the inability of many algorithms to account for amplitude correlations in the non-zero rows of $\boldsymbol{X}$.[2] Incorporating this temporal correlation structure is crucial, not only because many real-world signals possess such structure, but because the performance of MMV algorithms is particularly sensitive to this structure [13, 14, 35, 43, 49]. The second deficiency is that of computational complexity: while Bayesian MMV algorithms appear to offer the strongest recovery performance, it comes at the cost of increased complexity relative to simpler schemes, such as those based on greedy pursuit. For high-dimensional datasets, the complexity of Bayesian techniques may prohibit their application.

[2]Notable exceptions include [44], [41], and [14], which explicitly model amplitude correlations.

Our goal is to develop an MMV algorithm that offers the best of both worlds, combining the recovery performance of Bayesian techniques, even in the presence of substantial amplitude correlation and apriori unknown signal statistics, with the linear complexity scaling of greedy pursuit methods. Aiding us in meeting our goal is a powerful algorithmic framework known as *approximate message passing* (AMP), first proposed by Donoho et al. for the SMV CS problem [25]. In its early SMV formulations, AMP was shown to perform rapid and highly accurate probabilistic inference on models with known i.i.d. signal and noise priors, and i.i.d. random $\boldsymbol{A}$ matrices [25, 26]. More recently, AMP was extended to the block-sparse SMV problem under similar conditions [50]. While this block-sparse SMV AMP does solve a simple version of the MMV problem via the formulation (A.1), it does not account for intra-block amplitude correlation (i.e., temporal correlation in the MMV model). Recently, Kim et al. proposed an AMP-based MMV algorithm that does exploit temporal amplitude correlation [44]. However, their approach requires knowledge of the signal and noise statistics (e.g., sparsity, power, correlation) and uses matrix inversions at each iteration, implying a complexity that grows superlinearly in the problem dimensions.

In this chapter, we propose an AMP-based MMV algorithm (henceforth referred to as AMP-MMV) that exploits temporal amplitude correlation and learns the signal and noise statistics directly from the data, all while maintaining a computational complexity that grows linearly in the problem dimensions. In addition, AMP-MMV can easily accommodate time-varying measurement matrices $\boldsymbol{A}^{(t)}$, implicit measurement operators (e.g., FFT-based $\boldsymbol{A}$), and complex-valued quantities. (These latter scenarios occur in, e.g., digital communication [51] and pMRI [52].) The key to our approach lies in combining the "turbo AMP" framework of [53], where the usual

AMP factor graph is augmented with additional hidden variable nodes and inference is performed on the augmented factor graph, with an EM-based approach to hyperparameter learning. Details are provided in Sections 2.2, 2.4, and 2.5.

In Section 2.6, we present a detailed numerical study of AMP-MMV that includes a comparison against three state-of-the-art MMV algorithms. In order to establish an absolute performance benchmark, in Section 2.3 we describe a tight, oracle-aided performance lower bound that is realized through a support-aware Kalman smoother (SKS). To the best of our knowledge, our numerical study is the first in the MMV literature to use the SKS as a benchmark. Our numerical study demonstrates that AMP-MMV performs near this oracle performance bound under a wide range of problem settings, and that AMP-MMV is especially suitable for application to high-dimensional problems. In what represents a less-explored direction for the MMV problem, we also explore the effects of measurement matrix time-variation (cf. [33]). Our results show that measurement matrix time-variation can significantly improve reconstruction performance and thus we advocate the use of time-varying measurement operators whenever possible.

2.1.1 Notation

Boldfaced lower-case letters, e.g., $\boldsymbol{a}$, denote vectors, while boldfaced upper-case letters, e.g., $\boldsymbol{A}$, denote matrices. The letter t is strictly used to index a timestep, $t = 1, 2, \ldots, T$, the letter n is strictly used to index the coefficients of a signal, $n = 1, \ldots, N$, and the letter m is strictly used to index the measurements, $m = 1, \ldots, M$. The superscript $^{(t)}$ indicates a timestep-dependent quantity, while a superscript without parentheses, e.g., ^{k}, indicates a quantity whose value changes according to some algorithmic iteration index. Subscripted variables such as $x_n^{(t)}$ are used to denote the n^{th} element of the vector $\boldsymbol{x}^{(t)}$, while set subscript notation, e.g., $\boldsymbol{x}_{\mathcal{S}}^{(t)}$, denotes the

sub-vector of $\boldsymbol{x}^{(t)}$ consisting of indices contained in $\mathcal{S}$. The m^{th} row of the matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{a}_m^{\mathsf{T}}$, and the transpose (conjugate transpose) by $\boldsymbol{A}^{\mathsf{T}}$ ($\boldsymbol{A}^{\mathsf{H}}$). An M-by-M identity matrix is denoted by $\boldsymbol{I}_M$, a length-N vector of ones is given by $\mathbf{1}_N$ and $\boldsymbol{\mathcal{D}}(\boldsymbol{a})$ designates a diagonal matrix whose diagonal entries are given by the elements of the vector $\boldsymbol{a}$. Finally, $\mathcal{CN}(\boldsymbol{a}; \boldsymbol{b}, \boldsymbol{C})$ refers to the complex normal distribution that is a function of the vector $\boldsymbol{a}$, with mean $\boldsymbol{b}$ and covariance matrix $\boldsymbol{C}$.

2.2 Signal Model

In this section, we elaborate on the signal model outlined in Section 2.1, and make precise our modeling assumptions. Our signal model, as well as our algorithm, will be presented in the context of complex-valued signals, but can be easily modified to accommodate real-valued signals.

As noted in Section 2.1, we consider the linear measurement model (2.1), in which the signal $\boldsymbol{x}^{(t)} \in \mathbb{C}^N$ at timestep t is observed as $\boldsymbol{y}^{(t)} \in \mathbb{C}^M$ through the linear operator $\boldsymbol{A} \in \mathbb{C}^{M \times N}$. We assume $\boldsymbol{e}^{(t)} \sim \mathcal{CN}(\mathbf{0}, \sigma_e^2 \boldsymbol{I}_M)$ is circularly symmetric complex white Gaussian noise. We use $\mathcal{S} \triangleq \{n | x_n^{(t)} \neq 0\}$ to denote the indices of the time-invariant support of the signal, which is assumed to be suitably sparse, i.e., $|\mathcal{S}| \leq M$.[3]

Our approach to specifying a prior distribution for the signal, $p(\{\boldsymbol{x}^{(t)}\}_{t=1}^T)$, is motivated by a desire to separate the support, $\mathcal{S}$, from the amplitudes of the non-zero, or "active," coefficients. To accomplish this, we decompose each coefficient $x_n^{(t)}$

[3] If the signal being recovered is not itself sparse, it is assumed that there exists a known basis, incoherent with the measurement matrix, in which the signal possesses a sparse representation. Without loss of generality, we will assume the underlying signal is sparse in the canonical basis.

as the product of two hidden variables:

$$x_n^{(t)} = s_n \cdot \theta_n^{(t)} \quad \Leftrightarrow \quad p(x_n^{(t)}|s_n, \theta_n^{(t)}) = \begin{cases} \delta(x_n^{(t)} - \theta_n^{(t)}), & s_n = 1, \\ \delta(x_n^{(t)}), & s_n = 0, \end{cases} \tag{2.4}$$

where $s_n \in \{0, 1\}$ is a binary variable that indicates support set membership, $\theta_n^{(t)} \in \mathbb{C}$ is a variable that provides the amplitude of coefficient $x_n^{(t)}$, and $\delta(\cdot)$ is the Dirac delta function. When $s_n = 0$, $x_n^{(t)} = 0$ and $n \notin \mathcal{S}$, and when $s_n = 1$, $x_n^{(t)} = \theta_n^{(t)}$ and $n \in \mathcal{S}$. To model the sparsity of the signal, we treat each s_n as a Bernoulli random variable with $\Pr\{s_n = 1\} \triangleq \lambda_n < 1$.

In order to model the temporal correlation of signal amplitudes, we treat the evolution of amplitudes over time as stationary first-order Gauss-Markov random processes. Specifically, we assume that $\theta_n^{(t)}$ evolves according to the following linear dynamical system model:

$$\theta_n^{(t)} = (1 - \alpha)(\theta_n^{(t-1)} - \zeta) + \alpha w_n^{(t)} + \zeta, \tag{2.5}$$

where $\zeta \in \mathbb{C}$ is the mean of the amplitude process, $w_n^{(t)} \sim \mathcal{CN}(0, \rho)$ is a circularly symmetric white Gaussian perturbation process, and $\alpha \in [0, 1]$ is a scalar that controls the correlation of $\theta_n^{(t)}$ across time. At one extreme, $\alpha = 0$, the random process is perfectly correlated ($\theta_n^{(t)} = \theta_n^{(t-1)}$), while at the other extreme, $\alpha = 1$, the amplitudes evolve independently over time. Note that the binary support vector, $\boldsymbol{s}$, is independent of the amplitude random process, $\{\boldsymbol{\theta}^{(t)}\}_{t=1}^T$, which implies that there are hidden amplitude "trajectories", $\{\theta_n^{(t)}\}_{t=1}^T$, associated with inactive coefficients. Consequently, $\theta_n^{(t)}$ should be thought of as the conditional amplitude of $x_n^{(t)}$, conditioned on $s_n = 1$.

Under our model, the prior distribution of any signal coefficient, $x_n^{(t)}$, is a Bernoulli-Gaussian or "spike-and-slab" distribution:

$$p(x_n^{(t)}) = (1-\lambda_n)\delta\big(x_n^{(t)}\big) + \lambda_n \mathcal{CN}\big(x_n^{(t)}; \zeta, \sigma^2\big), \tag{2.6}$$

where $\sigma^2 \triangleq \frac{\alpha\rho}{2-\alpha}$ is the steady-state variance of $\theta_n^{(t)}$. We note that when $\lambda_n < 1$, (3.4) is an effective sparsity-promoting prior due to the point mass at $x_n^{(t)} = 0$.

2.3 The Support-Aware Kalman Smoother

Prior to describing AMP-MMV in detail, we first motivate the type of inference we wish to perform. Suppose for a moment that we are interested in obtaining a minimum mean square error (MMSE) estimate of $\{\boldsymbol{x}^{(t)}\}_{t=1}^T$, and that we have access to an oracle who can provide us with the support, $\mathcal{S}$. With this knowledge, we can concentrate solely on estimating $\{\boldsymbol{\theta}^{(t)}\}_{t=1}^T$, since, conditioned on $\mathcal{S}$, an MMSE estimate of $\{\boldsymbol{\theta}^{(t)}\}_{t=1}^T$ can provide an MMSE estimate of $\{\boldsymbol{x}^{(t)}\}_{t=1}^T$. For the linear dynamical system of (3.3), the support-aware Kalman smoother (SKS) provides the appropriate oracle-aided MMSE estimator of $\{\boldsymbol{\theta}^{(t)}\}_{t=1}^T$ [54]. The state-space model used by the SKS is:

$$\boldsymbol{\theta}^{(t)} = (1-\alpha)\boldsymbol{\theta}^{(t-1)} + \alpha\zeta\mathbf{1}_N + \alpha\boldsymbol{w}^{(t)}, \tag{2.7}$$

$$\boldsymbol{y}^{(t)} = \boldsymbol{A}\boldsymbol{\mathcal{D}}(\boldsymbol{s})\boldsymbol{\theta}^{(t)} + \boldsymbol{e}^{(t)}, \tag{2.8}$$

where $\boldsymbol{s}$ is the binary support vector associated with $\mathcal{S}$. If $\hat{\boldsymbol{\theta}}^{(t)}$ is the MMSE estimate returned by the SKS, then an MMSE estimate of $\boldsymbol{x}^{(t)}$ is given by $\hat{\boldsymbol{x}}^{(t)} = \boldsymbol{\mathcal{D}}(\boldsymbol{s})\hat{\boldsymbol{\theta}}^{(t)}$.

The state-space model (2.7), (2.8) provides a useful interpretation of our signal model. In the context of Kalman smoothing, the state vector $\boldsymbol{\theta}^{(t)}$ is only partially

observable (due to the action of $\boldsymbol{\mathcal{D}}(\boldsymbol{s})$ in (2.8)). Since $\boldsymbol{\mathcal{D}}(\boldsymbol{s})\boldsymbol{\theta}^{(t)} = \boldsymbol{x}^{(t)}$, noisy linear measurements of $\boldsymbol{x}^{(t)}$ are used to infer the state $\boldsymbol{\theta}^{(t)}$. However, since only those $\theta_n^{(t)}$ for which $n \in \mathcal{S}$ are observable, and thus identifiable, they are the only ones whose posterior distributions will be meaningful.

Since the SKS performs optimal MMSE estimation, given knowledge of the true signal support, it provides a useful lower bound on the achievable performance of any support-agnostic Bayesian algorithm that aims to perform MMSE estimation of $\{\boldsymbol{x}^{(t)}\}_{t=1}^{T}$.

2.4 The AMP-MMV Algorithm

In Section 2.2, we decomposed each signal coefficient, $x_n^{(t)}$, as the product of a binary support variable, s_n, and an amplitude variable, $\theta_n^{(t)}$. We now develop an algorithm that infers a marginal posterior distribution on each variable, enabling both soft estimation and soft support detection.

The statistical structure of the signal model from Section 2.2 becomes apparent from a factorization of the posterior joint pdf of all random variables. Recalling from (A.1) the definitions of $\bar{\boldsymbol{y}}$ and $\bar{\boldsymbol{x}}$, and defining $\bar{\boldsymbol{\theta}}$ similarly, the posterior joint distribution factors as follows:

$$p(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\theta}}, \boldsymbol{s}|\bar{\boldsymbol{y}}) \propto \prod_{t=1}^{T} \left(\prod_{m=1}^{M} p(y_m^{(t)}|\boldsymbol{x}^{(t)}) \prod_{n=1}^{N} p(x_n^{(t)}|\theta_n^{(t)}, s_n) p(\theta_n^{(t)}|\theta_n^{(t-1)}) \right) \prod_{n=1}^{N} p(s_n), \quad (2.9)$$

where $\propto$ indicates equality up to a normalizing constant, and $p(\theta_n^{(1)}|\theta_n^{(0)}) \triangleq p(\theta_n^{(1)})$. A convenient graphical representation of this decomposition is given by a *factor graph* [55], which is an undirected bipartite graph that connects the pdf "factors" of (3.5) with the variables that make up their arguments. The factor graph for the decomposition of (3.5) is shown in Fig. 2.1. The *factor nodes* are denoted by filled

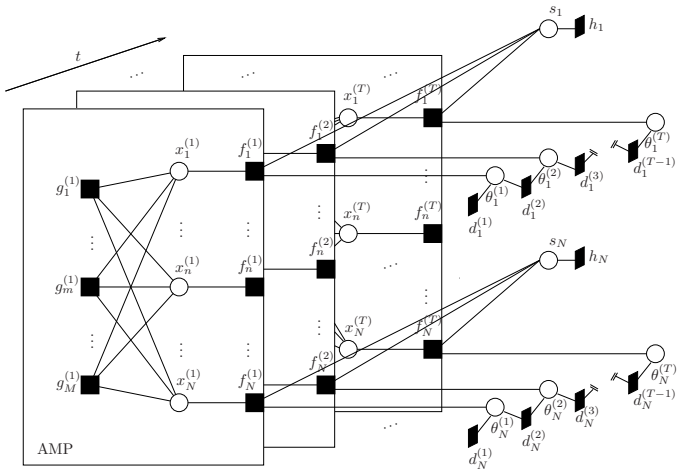


Figure 2.1: Factor graph representation of the $p(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\theta}}, \boldsymbol{s}|\bar{\boldsymbol{y}})$ decomposition in (3.5).

squares, while the *variable nodes* are denoted by circles. In the figure, the signal variable nodes at timestep t, $\{x_n^{(t)}\}_{n=1}^N$, are depicted as lying in a plane, or "frame", with successive frames stacked one after another. Since during inference the measurements $\{y_m^{(t)}\}$ are known observations and not random variables, they do not appear explicitly in the factor graph. The connection between the frames occurs through the amplitude and support indicator variables, providing a graphical representation of the temporal correlation in the signal. For visual clarity, these $\{\theta_n^{(t)}\}_{t=1}^T$ and s_n variable nodes have been removed from the graph for the intermediate index n, but should in fact be present at every index $n = 1, \ldots, N$.

The factor nodes in Fig. 2.1 have all been assigned alphabetic labels; the correspondence between these labels and the distributions they represent, as well as the functional form of each distribution, is presented in Table 3.1.

A natural approach to performing statistical inference on a signal model that possesses a convenient factor graph representation is through a message passing algorithm

Factor	Distribution	Functional Form
$g_m^{(t)}\big(\boldsymbol{x}^{(t)}\big)$	$p\big(y_m^{(t)}\big\vert\boldsymbol{x}^{(t)}\big)$	$\mathcal{CN}\big(y_m^{(t)};\boldsymbol{a}_m^{\mathsf{T}}\boldsymbol{x}^{(t)},\sigma_e^2\big)$
$f_n^{(t)}\big(x_n^{(t)},s_n,\theta_n^{(t)}\big)$	$p\big(x_n^{(t)}\big\vert s_n,\theta_n^{(t)}\big)$	$\delta\big(x_n^{(t)}-s_n\theta_n^{(t)}\big)$
$h_n\big(s_n\big)$	$p\big(s_n\big)$	$\big(1-\lambda_n\big)^{(1-s_n)}\big(\lambda_n\big)^{s_n}$
$d_n^{(1)}\big(\theta_n^{(1)}\big)$	$p\big(\theta_n^{(1)}\big)$	$\mathcal{CN}\big(\theta_n^{(1)};\zeta,\sigma^2\big)$
$d_n^{(t)}\big(\theta_n^{(t)},\theta_n^{(t-1)}\big)$	$p\big(\theta_n^{(t)}\big\vert\theta_n^{(t-1)}\big)$	$\mathcal{CN}\big(\theta_n^{(t)};(1-\alpha)\theta_n^{(t-1)}+\alpha\zeta,\alpha^2\rho\big)$

Table 2.1: The factors, underlying distributions, and functional forms associated with the signal model of Section 2.2.

known as belief propagation [56]. In belief propagation, the messages exchanged between connected nodes of the graph represent probability distributions. In cycle-free graphs, belief propagation can be viewed as an instance of the sum-product algorithm [55], allowing one to obtain an exact posterior marginal distribution for each unobserved variable, given a collection of observed variables. When the factor graph contains cycles, the same rules that define the sum-product algorithm can still be applied, however convergence is no longer guaranteed [55]. Despite this, there exist many problems to which loopy belief propagation [28] has been successfully applied, including inference on Markov random fields [57], LDPC decoding [58], and compressed sensing [25, 27, 29, 53, 59].

We now proceed with a high-level description of AMP-MMV, an algorithm that follows the sum-product methodology while leveraging recent advances in message approximation [25]. In what follows, we use $\nu_{a\to b}(\cdot)$ to denote a message that is passed from node a to a connected node b.

2.4.1 Message Scheduling

Since the factor graph of Fig. 2.1 contains many cycles there are a number of valid ways to schedule, or sequence, the messages that are exchanged in the graph. We will

describe two message passing schedules that empirically provide good convergence behavior and straightforward implementation. We refer to these two schedules as the *parallel message schedule* and the *serial message schedule*. In both cases, messages are first initialized to agnostic values, and then iteratively exchanged throughout the graph according to the chosen schedule until either convergence occurs, or a maximum number of allowable iterations is reached.

Conceptually, both message schedules can be decomposed into four distinct phases, differing only in which messages are initialized and the order in which the phases are sequenced. We label each phase using the mnemonics **(into)**, **(within)**, **(out)**, and **(across)**. In phase **(into)**, messages are passed from the s_n and $\theta_n^{(t)}$ variable nodes *into* frame t. Loosely speaking, these messages convey current beliefs about the values of $\boldsymbol{s}$ and $\boldsymbol{\theta}^{(t)}$. In phase **(within)**, messages are exchanged *within* frame t, producing an estimate of $\boldsymbol{x}^{(t)}$ using the current beliefs about $\boldsymbol{s}$ and $\boldsymbol{\theta}^{(t)}$ together with the available measurements $\boldsymbol{y}^{(t)}$. In phase **(out)**, the estimate of $\boldsymbol{x}^{(t)}$ is used to refine the beliefs about $\boldsymbol{s}$ and $\boldsymbol{\theta}^{(t)}$ by passing messages *out* of frame t. Finally, in phase **(across)**, messages are sent from $\theta_n^{(t)}$ to either $\theta_n^{(t+1)}$ or $\theta_n^{(t-1)}$, thus conveying information *across* time about temporal correlation in the signal amplitudes.

The parallel message schedule begins by performing phase **(into)** in parallel for each frame $t = 1, \ldots, T$ simultaneously. Then, phase **(within)** is performed simultaneously for each frame, followed by phase **(out)**. Next, information about the amplitudes is exchanged between the different timesteps by performing phase **(across)** in the forward direction, i.e., messages are passed from $\theta_n^{(1)}$ to $\theta_n^{(2)}$, and then from $\theta_n^{(2)}$ to $\theta_n^{(3)}$, proceeding until $\theta_n^{(T)}$ is reached. Finally, phase **(across)** is performed in the backward direction, where messages are passed consecutively from $\theta_n^{(T)}$ down to $\theta_n^{(1)}$. At this point, a single iteration of AMP-MMV has been completed, and a

new iteration can commence starting with phase **(into)**. In this way, all of the available measurements, $\{\boldsymbol{y}^{(t)}\}_{t=1}^{T}$, are used to influence the recovery of the signal at each timestep.

The serial message schedule is similar to the parallel schedule except that it operates on frames in a sequential fashion, enabling causal processing of MMV signals. Beginning at the initial timestep, $t = 1$, the serial schedule first performs phase **(into)**, followed by phases **(within)** and **(out)**. Outgoing messages from the initial frame are then used in phase **(across)** to pass messages from $\theta_n^{(1)}$ to $\theta_n^{(2)}$. The messages arriving at $\theta_n^{(2)}$, along with updated beliefs about the value of $\boldsymbol{s}$, are used to initiate phase **(into)** at timestep $t = 2$. Phases **(within)** and **(out)** are performed for frame 2, followed by another round of phase **(across)**, with messages being passed forward to $\theta_n^{(3)}$. This procedure continues until phase **(out)** is completed at frame T. Until now, only causal information has been used in producing estimates of the signal. If the application permits smoothing, then message passing continues in a similar fashion, but with messages now propagating backward in time, i.e., messages are passed from $\theta_n^{(T)}$ to $\theta_n^{(T-1)}$, phases **(into)**, **(within)**, and **(out)** are performed at frame $T - 1$, and then messages move from $\theta_n^{(T-1)}$ to $\theta_n^{(T-2)}$. The process continues until messages arrive at $\theta_n^{(1)}$, at which point a single *forward/backward pass* has been completed. We complete multiple such passes, resulting in a smoothed estimate of the signal.

2.4.2 Implementing the Message Passes

Space constraints prohibit us from providing a full derivation of all the messages that are exchanged through the factor graph of Fig. 2.1. Most messages can be derived by straightforward application of the rules of the sum-product algorithm. Therefore, in this sub-section we will restrict our attention to a handful of messages in the **(within)**

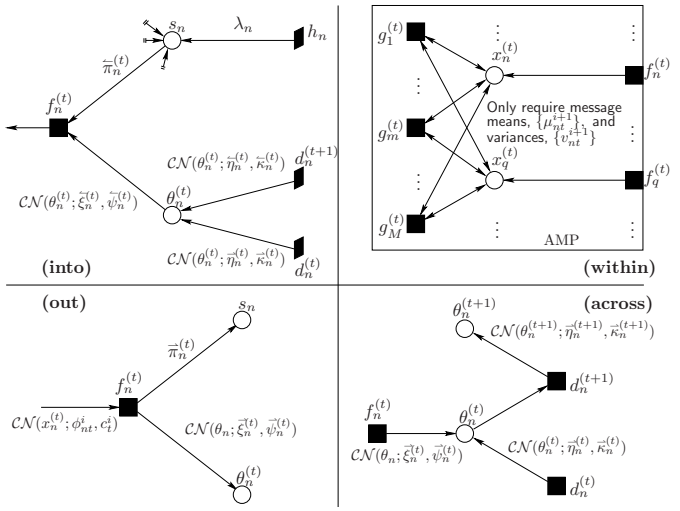


Figure 2.2: A summary of the four message passing phases, including message notation and form.

and **(out)** phases whose implementation requires a departure from the sum-product rules for one reason or another.

To aid our discussion, in Fig. 2.2 we summarize each of the four phases, focusing primarily on a single coefficient index n at some intermediate frame t. Arrows indicate the direction that messages are moving, and only those nodes and edges participating in a particular phase are shown in that phase. For the **(across)** phase we show messages being passed forward in time, and omit a graphic for the corresponding backwards pass. The figure also introduces the notation that we adopt for the different variables that serve to parameterize the messages. Certain variables, e.g., $\vec{\eta}_n^{(t)}$ and $\overleftarrow{\eta}_n^{(t)}$, are accented with directional arrows. This is to distinguish variables associated with messages moving in one direction from those associated with messages moving in another. For Bernoulli message pdfs, we show only the nonzero probability, e.g., $\lambda_n = \nu_{h_n \to s_n}(s_n = 1)$.

Phase **(within)** entails using the messages transmitted from s_n and $\theta_n^{(t)}$ to $f_n^{(t)}$

to compute the messages that pass between $x_n^{(t)}$ and the $\{g_m^{(t)}\}$ nodes. Inspection of Fig. 2.2 reveals a dense interconnection between the $\{x_n^{(t)}\}$ and $\{g_m^{(t)}\}$ nodes. As a consequence, applying the standard sum-product rules to compute the $\nu_{g_m^{(t)}\to x_n^{(t)}}(\cdot)$ messages would result in an algorithm that required the evaluation of multi-dimensional integrals that grew exponentially in number in both N and M. Since we are strongly motivated to apply AMP-MMV to high-dimensional problems, this approach is clearly infeasible. Instead, we turn to a recently developed algorithm known as *approximate message passing* (AMP).

AMP was originally proposed by Donoho et al. [25] as a message passing algorithm designed to solve the noiseless SMV CS problem known as Basis Pursuit ($\min \|\boldsymbol{x}\|_1$ s.t. $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$), and was subsequently extended [26] to support MMSE estimation under white-Gaussian-noise-corrupted observations and generic signal priors of the form $p(\boldsymbol{x}) = \prod p(x_n)$ through an approximation of the sum-product algorithm. In both cases, the associated factor graph looks identical to that of the **(within)** segment of Fig. 2.2. Conventional wisdom holds that loopy belief propagation only works well when the factor graph is locally tree-like. For general, non-sparse $\boldsymbol{A}$ matrices, the **(within)** graph will clearly not possess this property, due to the many short cycles between the $x_n^{(t)}$ and $g_m^{(t)}$ nodes. Reasoning differently, Donoho et al. showed that the density of connections could prove beneficial, if properly exploited.

In particular, central limit theorem arguments suggest that the messages propagated from the g_m nodes to the x_n nodes under the sum-product algorithm can be well-approximated as Gaussian when the problem dimensionality is sufficiently high. Moreover, the computation of these Gaussian-approximated messages only requires knowledge of the mean and variance of the sum-product messages from the x_n to the g_m nodes. Finally, when $|A_{mn}|^2$ scales as $\mathcal{O}(1/M)$ for all (m, n), the differences between the variances of the messages emitted by the x_n nodes vanish as M grows

large, as do those of the g_m nodes when N grows large, allowing each to be approximated by a single, common variance. Together, these sum-product approximations yield an iterative thresholding algorithm with a particular first-order correction term that ensures both Gaussianity and independence in the residual error vector over the iterations. The complexity of this iterative thresholding algorithm is dominated by a single multiplication by $\boldsymbol{A}$ and $\boldsymbol{A}^{\mathsf{H}}$ per iteration, implying a per-iteration computational cost of $\mathcal{O}(MN)$ flops. Furthermore, the state-evolution equation that governs the transient behavior of AMP shows that the number of required iterations does not scale with either M or N, implying that the total complexity is itself $\mathcal{O}(MN)$ flops. For the interested reader, in Appendix A, we provide additional background material on the AMP algorithm.

AMP's suitability for the MMV problem stems from several considerations. First, AMP's probabilistic construction, coupled with its message passing implementation, makes it well-suited for incorporation as a subroutine within a larger message passing algorithm. In the MMV problem it is clear that $p(\bar{\boldsymbol{x}}) \neq \prod p(x_n^{(t)})$ due to the joint sparsity and amplitude correlation structure, and therefore AMP does not appear to be directly applicable. Fortunately, by modeling this structure through the hidden variables $\boldsymbol{s}$ and $\bar{\boldsymbol{\theta}}$, we can exploit the conditional independence of the signal coefficients: $p(\bar{\boldsymbol{x}}|\boldsymbol{s}, \bar{\boldsymbol{\theta}}) = \prod p(x_n^{(t)}|s_n, \theta_n^{(t)})$.

By viewing $\nu_{f_n^{(t)} \to x_n^{(t)}}(\cdot)$ as a "local prior"[4] for $x_n^{(t)}$, we can readily apply an off-the-shelf AMP algorithm (e.g., [26, 27]) as a means of performing the message passes within the portions of the factor graph enclosed within the frames of Fig. 2.1. The use of AMP with decoupled local priors within a larger message passing algorithm that

[4]The AMP algorithm is conventionally run with static, i.i.d. priors for each signal coefficient. When utilized as a sub-component of a larger message passing algorithm on an expanded factor graph, the signal priors (from AMP's perspective) will be changing in response to messages from the rest of the factor graph. We refer to these changing AMP priors as *local priors*.

accounts for statistical dependencies between signal coefficients was first proposed in [53], and further studied in [15, 16, 32, 60, 61]. Here, we exploit this powerful "turbo" inference approach to account for the strong temporal dependencies inherent in the MMV problem.

The local prior on $x_n^{(t)}$ given the current belief about the hidden variables s_n and $\theta_n^{(t)}$ assumes the Bernoulli-Gaussian form

$$\nu_{f_n^{(t)} \to x_n^{(t)}}(x_n^{(t)}) = (1 - \overleftarrow{\pi}_n^{(t)})\delta(x_n^{(t)}) + \overleftarrow{\pi}_n^{(t)} \mathcal{CN}(x_n^{(t)}; \overleftarrow{\xi}_n^{(t)}, \overleftarrow{\psi}_n^{(t)}). \tag{2.10}$$

This local prior determines the AMP soft-thresholding functions defined in (D5) - (D8) of Table 2.2. The derivation of these thresholding functions closely follows those outlined in [53], which considered the special case of a zero-mean Bernoulli-Gaussian prior.

Beyond the ease with which AMP is included into the larger message passing algorithm, a second factor that favors using AMP is the tremendous computational efficiency it imparts on high-dimensional problems. Using AMP to perform the most computationally intensive message passes enables AMP-MMV to attain a linear complexity scaling in all problem dimensions. To see why this is the case, note that the **(into)**, **(out)**, and **(across)** steps can be executed in $\mathcal{O}(N)$ flops/timestep, while AMP allows the **(within)** step to be executed in $\mathcal{O}(MN)$ flops/timestep (see (A17) - (A21) of Table 2.2). Since these four steps are executed $\mathcal{O}(T)$ times per AMP-MMV iteration for both the serial and parallel message schedules, it follows that AMP-MMV's overall complexity is $\mathcal{O}(TMN)$.[5]

A third appealing feature of AMP is that it is theoretically well-grounded; a recent

[5]The primary computational burden of executing AMP-MMV involves performing matrix-vector products with $\boldsymbol{A}$ and $\boldsymbol{A}^{\mathsf{H}}$, allowing it to be easily applied in problems where the measurement matrix is never stored explicitly, but rather is implemented implicitly through subroutines. Fast implicit $\boldsymbol{A}$ operators can provide significant computational savings in high-dimensional problems;

% Define soft-thresholding functions:

$$F_{nt}(\phi;c) \triangleq (1+\gamma_{nt}(\phi;c))^{-1}\Big(\frac{\overleftarrow{\psi}_n^{(t)}\phi+\overleftarrow{\xi}_n^{(t)}c}{\overleftarrow{\psi}_n^{(t)}+c}\Big) \tag{D1}$$

$$G_{nt}(\phi;c) \triangleq (1+\gamma_{nt}(\phi;c))^{-1}\Big(\frac{\overleftarrow{\psi}_n^{(t)}c}{\overleftarrow{\psi}_n^{(t)}+c}\Big) + \gamma_{nt}(\phi;c)|F_n(\phi;c)|^2 \tag{D2}$$

$$F'_{nt}(\phi;c) \triangleq \tfrac{\partial}{\partial\phi}F_{nt}(\phi,c) = \tfrac{1}{c}G_{nt}(\phi;c) \tag{D3}$$

$$\gamma_{nt}(\phi;c) \triangleq \Big(\frac{1-\overleftarrow{\pi}_n^{(t)}}{\overleftarrow{\pi}_n^{(t)}}\Big)\Big(\frac{\overleftarrow{\psi}_n^{(t)}+c}{c}\Big) \times \exp\Big(-\Big[\frac{\overleftarrow{\psi}_n^{(t)}|\phi|^2+\overleftarrow{\xi}_n^{(t)*}c\phi+\overleftarrow{\xi}_n^{(t)}c\phi^*-c|\overleftarrow{\xi}_n^{(t)}|^2}{c(\overleftarrow{\psi}_n^{(t)}+c)}\Big]\Big) \tag{D4}$$

% Begin passing messages...

for $t = 1,\dots,T$, $\forall n$:

% Execute the **(into)** phase...

$$\overleftarrow{\pi}_n^{(t)} = \frac{\lambda_n\cdot\prod_{t'\neq t}\overrightarrow{\pi}_n^{(t')}}{(1-\lambda_n)\cdot\prod_{t'\neq t}(1-\overrightarrow{\pi}_n^{(t')})+\lambda_n\cdot\prod_{t'\neq t}\overrightarrow{\pi}_n^{(t')}} \tag{A1}$$

$$\overleftarrow{\psi}_n^{(t)} = \frac{\overrightarrow{\kappa}_n^{(t)}\cdot\overleftarrow{\kappa}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}+\overleftarrow{\kappa}_n^{(t)}} \tag{A2}$$

$$\overleftarrow{\xi}_n^{(t)} = \overleftarrow{\psi}_n^{(t)}\cdot\Big(\frac{\overrightarrow{\eta}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}}+\frac{\overleftarrow{\eta}_n^{(t)}}{\overleftarrow{\kappa}_n^{(t)}}\Big) \tag{A3}$$

% Initialize AMP-related variables...

$\forall m: z_{mt}^1 = y_m^{(t)}, \forall n: \mu_{nt}^1 = 0,$ and $c_t^1 = 100\cdot\sum_{n=1}^N\psi_n^{(t)}$

% Execute the **(within)** phase using AMP...

for $i = 1,\dots,I$, $\forall n,m$:

$$\phi_{nt}^i = \textstyle\sum_{m=1}^M A_{mn}^* z_{mt}^i + \mu_{nt}^i \tag{A4}$$

$$\mu_{nt}^{i+1} = F_{nt}(\phi_{nt}^i;c_t^i) \tag{A5}$$

$$v_{nt}^{i+1} = G_{nt}(\phi_{nt}^i;c_t^i) \tag{A6}$$

$$c_t^{i+1} = \sigma_e^2 + \tfrac{1}{M}\textstyle\sum_{n=1}^N v_{nt}^{i+1} \tag{A7}$$

$$z_{mt}^{i+1} = y_m^{(t)} - \boldsymbol{a}_m^{\mathsf{T}}\boldsymbol{\mu}_t^{i+1} + \tfrac{z_{mt}^i}{M}\textstyle\sum_{n=1}^N F'_{nt}(\phi_{nt}^i;c_t^i) \tag{A8}$$

end

$$\hat{x}_n^{(t)} = \mu_{nt}^{I+1} \qquad \text{\% Store current estimate of } x_n^{(t)} \tag{A9}$$

% Execute the **(out)** phase...

$$\overrightarrow{\pi}_n^{(t)} = \Big(1+\Big(\frac{\overleftarrow{\pi}_n^{(t)}}{1-\overleftarrow{\pi}_n^{(t)}}\Big)\gamma_{nt}(\phi_{nt}^I;c_t^{I+1})\Big)^{-1} \tag{A10}$$

$$(\overrightarrow{\xi}_n^{(t)},\overrightarrow{\psi}_n^{(t)}) = \text{taylor\_approx}(\overrightarrow{\pi}_n^{(t)},\phi_{nt}^I,c_t^I) \tag{A11}$$

% Execute the **(across)** phase from $\theta_n^{(t)}$ to $\theta_n^{(t+1)}$...

$$\overrightarrow{\eta}_n^{(t+1)} = (1-\alpha)\Big(\frac{\overrightarrow{\kappa}_n^{(t)}\overrightarrow{\psi}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}+\overrightarrow{\psi}_n^{(t)}}\Big)\Big(\frac{\overrightarrow{\eta}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}}+\frac{\overrightarrow{\xi}_n^{(t)}}{\overrightarrow{\psi}_n^{(t)}}\Big)+\alpha\zeta \tag{A12}$$

$$\overrightarrow{\kappa}_n^{(t+1)} = (1-\alpha)^2\Big(\frac{\overrightarrow{\kappa}_n^{(t)}\overrightarrow{\psi}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}+\overrightarrow{\psi}_n^{(t)}}\Big)+\alpha^2\rho \tag{A13}$$

end

Table 2.2: Message update equations for executing a single forward pass using the serial message schedule.

analysis [29] shows that, for Gaussian $\boldsymbol{A}$ in the large-system limit (i.e., M, $N\to\infty$ with M/N fixed), the behavior of AMP is governed by a state evolution whose fixed points, when unique, correspond to MMSE-optimal signal estimates.

implementing a Fourier transform as a fast Fourier transform (FFT) subroutine, for example, would drop AMP-MMV's complexity from $\mathcal{O}(TMN)$ to $\mathcal{O}(TN\log_2 N)$.

After employing AMP to manage the message passing between the $\left\{x_n^{(t)}\right\}_{n=1}^N$ and $\left\{g_m^{(t)}\right\}_{m=1}^M$ nodes in step **(within)**, messages must be propagated out of the dashed AMP box of frame t (step **(out)**) and either forward or backward in time (step **(across)**). While step **(across)** simply requires a straightforward application of the sum-product message computation rules, step **(out)** imposes several difficulties which we must address. For the remainder of this discussion, we focus on a novel approximation scheme for specifying the message $\nu_{f_n^{(t)}\to\theta_n^{(t)}}(\cdot)$. Our objective is to arrive at a message approximation that introduces negligible error while still leading to a computationally efficient algorithm. A Gaussian message approximation is a natural choice, given the marginally Gaussian distribution of $\theta_n^{(t)}$. As we shall soon see, it is also a highly justifiable choice.

A routine application of the sum-product rules to the $f_n^{(t)}$-to-$\theta_n^{(t)}$ message would produce the following expression:

$$\nu^{\text{exact}}_{f_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)}) \triangleq (1-\overleftarrow{\pi}_n^{(t)})\mathcal{CN}(0;\phi_{nt}^i,c_t^i)+\overleftarrow{\pi}_n^{(t)}\mathcal{CN}(\theta_n^{(t)};\phi_{nt}^i,c_t^i). \tag{2.11}$$

Unfortunately, the term $\mathcal{CN}(0;\phi_{nt}^i,c_t^i)$ prevents us from normalizing $\nu^{\text{exact}}_{f_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)})$, because it is constant with respect to $\theta_n^{(t)}$. Therefore, the distribution on $\theta_n^{(t)}$ represented by (C.32) is improper. To provide intuition into why this is the case, it is helpful to think of $\nu_{f_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)})$ as a message that conveys information about the value of $\theta_n^{(t)}$ based on the values of $x_n^{(t)}$ and $s_n^{(t)}$. If $s_n^{(t)}=0$, then by (3.2), $x_n^{(t)}=0$, thus making $\theta_n^{(t)}$ unobservable. The constant term in (C.32) reflects the uncertainty due to this unobservability through an infinitely broad, uninformative distribution for $\theta_n^{(t)}$.

To avoid an improper pdf, we modify how this message is derived by regarding our assumed signal model, in which $s_n^{(t)}\in\{0,1\}$, as a limiting case of the model with

$s_n^{(t)} \in \{\varepsilon, 1\}$ as $\varepsilon \to 0$. For any fixed positive ε, the resulting message $\nu_{f_n^{(t)} \to \theta_n^{(t)}}(\cdot)$ is proper, given by

$$\nu^{\text{mod}}_{f_n^{(t)} \to \theta_n^{(t)}}(\theta_n^{(t)}) = (1 - \Omega(\overleftarrow{\pi}_n^{(t)}))\ \mathcal{CN}(\theta_n^{(t)}; \tfrac{1}{\varepsilon}\phi_{nt}^i, \tfrac{1}{\varepsilon^2}c_t^i) + \Omega(\overleftarrow{\pi}_n^{(t)})\ \mathcal{CN}(\theta_n^{(t)}; \phi_{nt}^i, c_t^i), \quad (2.12)$$

where

$$\Omega(\pi) \triangleq \frac{\varepsilon^2 \pi}{(1-\pi) + \varepsilon^2 \pi}. \quad (2.13)$$

The pdf in (2.12) is that of a binary Gaussian mixture. If we consider $\varepsilon \ll 1$, the first mixture component is extremely broad, while the second is more "informative," with mean ϕ_n^i and variance c_n^i. The relative weight assigned to each component Gaussian is determined by the term $\Omega(\overleftarrow{\pi}_n^{(t)})$. Notice that the limit of this weighting term is the simple indicator function

$$\lim_{\varepsilon \to 0} \Omega(\pi) = \begin{cases} 0 & \text{if } 0 \leq \pi < 1, \\ 1 & \text{if } \pi = 1. \end{cases} \quad (2.14)$$

Since we cannot set $\varepsilon = 0$, we instead fix a small positive value, e.g., $\varepsilon = 10^{-7}$. In this case, (2.12) could then be used as the outgoing message. However, this presents a further difficulty: propagating a binary Gaussian mixture forward in time would lead to an exponential growth in the number of mixture components at subsequent timesteps. This difficulty is a familiar one in the context of switched linear dynamical systems based on conditional Gaussian models, since such models are not closed under marginalization [62]. To avoid the exponential growth in the number of mixture components, we collapse our binary Gaussian mixture to a single Gaussian component, an approach sometimes referred to as a Gaussian sum approximation [63, 64]. This can be justified by the fact that, for $\varepsilon \ll 1$, $\Omega(\cdot)$ behaves nearly like the indicator

function $(\vec{\xi},\ \vec{\psi}) =$ taylor_approx$(\overleftarrow{\pi},\ \phi,\ c)$	
% Define useful variables:	
$\mathfrak{a} \triangleq \varepsilon^2(1-\Omega(\overleftarrow{\pi}))$	(T1)
$\bar{\mathfrak{a}} \triangleq \Omega(\overleftarrow{\pi})$	(T2)
$\mathfrak{b} \triangleq \frac{\varepsilon^2}{c}\lvert(1-\frac{1}{\varepsilon})\phi\rvert^2$	(T3)
$\mathfrak{d}_r \triangleq -\frac{2\varepsilon^2}{c}(1-\frac{1}{\varepsilon})\mathfrak{Re}\{\phi\}$	(T4)
$\mathfrak{d}_i \triangleq -\frac{2\varepsilon^2}{c}(1-\frac{1}{\varepsilon})\mathfrak{Im}\{\phi\}$	(T5)
% Compute outputs:	
$\vec{\psi} = \frac{(\mathfrak{a}^2 e^{-\mathfrak{b}}+\mathfrak{a}\bar{\mathfrak{a}}+\bar{\mathfrak{a}}^2 e^{\mathfrak{b}})c}{\varepsilon^2\mathfrak{a}^2 e^{-\mathfrak{b}}+\mathfrak{a}\bar{\mathfrak{a}}(\varepsilon^2+1-\frac{1}{2}c\mathfrak{d}_r^2)+\bar{\mathfrak{a}}^2 e^{\mathfrak{b}}}$	(T6)
$\vec{\xi}_r = \phi_r - \frac{1}{2}\vec{\psi}\frac{-\mathfrak{a}e^{-\mathfrak{b}}\mathfrak{d}_r}{\mathfrak{a}e^{-\mathfrak{b}}+\bar{\mathfrak{a}}}$	(T7)
$\vec{\xi}_i = \phi_i - \frac{1}{2}\vec{\psi}\frac{-\mathfrak{a}e^{-\mathfrak{b}}\mathfrak{d}_i}{\mathfrak{a}e^{-\mathfrak{b}}+\bar{\mathfrak{a}}}$	(T8)
$\vec{\xi} = \vec{\xi}_r + j\vec{\xi}_i$	(T9)
return $(\vec{\xi},\ \vec{\psi})$	

Table 2.3: Pseudocode function for computing a single-Gaussian approximation of (2.12).

function in (C.35), in which case one of the two Gaussian components will typically have negligible mass.

To carry out the collapsing, we perform a second-order Taylor series approximation of $-\log \nu^{\text{mod}}_{f_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)})$ with respect to $\theta_n^{(t)}$ about the point ϕ_{nt}.[6] This provides the mean, $\vec{\xi}_n^{(t)}$, and variance, $\vec{\psi}_n^{(t)}$, of the single Gaussian that serves as $\nu_{f_n^{(t)}\to\theta_n^{(t)}}(\cdot)$. (See Fig. 2.2.) In Appendix B we summarize the Taylor approximation procedure, and in Table 2.3 provide the pseudocode function, taylor_approx, for computing $\vec{\xi}_n^{(t)}$ and $\vec{\psi}_n^{(t)}$.

With the exception of the messages discussed above, all the remaining messages can be derived using the standard sum-product algorithm rules [55]. For convenience, we summarize the results in Table 2.2, where we provide a pseudocode implementation of a single forward pass of AMP-MMV using the serial message schedule.

[6]For technical reasons, the Taylor series approximation is performed in $\mathbb{R}^2$ instead of $\mathbb{C}$.

2.5 Estimating the Model Parameters

The signal model of Section 2.2 depends on the sparsity parameters $\{\lambda_n\}_{n=1}^N$, amplitude parameters ζ, α, and ρ, and noise variance σ_e^2. While some of these parameters may be known accurately from prior information, it is likely that many will require tuning. To this end, we develop an expectation-maximization (EM) algorithm that couples with the message passing procedure described in Section 2.4.1 to provide a means of learning all of the model parameters while simultaneously estimating the signal $\bar{\boldsymbol{x}}$ and its support $\boldsymbol{s}$.

The EM algorithm [65] is an appealing choice for performing parameter estimation for two primary reasons. First and foremost, the EM algorithm is a well-studied and principled means of parameter estimation. At every EM iteration, the data likelihood function is guaranteed to increase until convergence to a local maximum of the likelihood function occurs [65]. For multimodal likelihood functions, local maxima will, in general, not coincide with the global maximum likelihood (ML) estimator, however a judicious initialization can help in ensuring the EM algorithm reaches the global maximum [66]. Second, the expectation step of the EM algorithm relies on quantities that have already been computed in the process of executing AMP-MMV. Ordinarily, this step constitutes the major computational burden of any EM algorithm, thus the fact that we can perform it essentially for free makes our EM procedure highly efficient.

We let $\Gamma \triangleq \{\lambda, \zeta, \alpha, \rho, \sigma_e^2\}$ denote the set of all model parameters, and let Γ^k denote the set of parameter estimates at the k^{th} EM iteration. Here we have assumed that the binary support indicator variables share a common activity probability, λ, i.e., $\Pr\{s_n = 1\} = \lambda \; \forall n$. The objective of the EM procedure is to find parameter estimates that maximize the data likelihood $p(\bar{\boldsymbol{y}}|\Gamma)$. Since it is often computationally intractable to perform this maximization, the EM algorithm incorporates additional

"hidden" data and iterates between two steps: *(i)* evaluating the conditional expectation of the log likelihood of the hidden data given the observed data, $\bar{\boldsymbol{y}}$, and the current estimates of the parameters, Γ^k, and *(ii)* maximizing this expected log likelihood with respect to the model parameters. For all parameters except σ_e^2 we use $\boldsymbol{s}$ and $\bar{\boldsymbol{\theta}}$ as the hidden data, while for σ_e^2 we use $\bar{\boldsymbol{x}}$.

For the first iteration of AMP-MMV, the model parameters are initialized based on either prior signal knowledge, or according to some heuristic criteria. Using these parameter values, AMP-MMV performs either a single iteration of the parallel message schedule, or a single forward/backward pass of the serial message schedule, as described in Section 2.4.1. Upon completing this first iteration, approximate marginal posterior distributions are available for each of the underlying random variables, e.g., $p(x_n^{(t)}|\bar{\boldsymbol{y}})$, $p(s_n|\bar{\boldsymbol{y}})$, and $p(\theta_n^{(t)}|\bar{\boldsymbol{y}})$. Additionally, belief propagation can provide pairwise joint posterior distributions, e.g., $p(\theta_n^{(t)}, \theta_n^{(t-1)}|\bar{\boldsymbol{y}})$, for any variable nodes connected by a common factor node [67]. With these marginal, and pairwise joint, posterior distributions, it is possible to perform the iterative expectation and maximization steps required to maximize $p(\bar{\boldsymbol{y}}|\Gamma)$ in closed-form. We adopt a Gauss-Seidel scheme, performing coordinate-wise maximization, e.g.,

$$\lambda^{k+1} = \underset{\lambda}{\operatorname{argmax}} \, \mathrm{E}_{\boldsymbol{s},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}} \left[\log p(\bar{\boldsymbol{y}}, \boldsymbol{s}, \bar{\boldsymbol{\theta}}; \lambda, \Gamma^k \backslash \{\lambda^k\}) \middle| \bar{\boldsymbol{y}}, \Gamma^k \right],$$

where k is the iteration index common to both AMP-MMV and the EM algorithm.

In Table 3.3 we provide the EM parameter update equations for our signal model. In practice, we found that the robustness and convergence behavior of our EM procedure were improved if we were selective about which parameters we updated on a given iteration. For example, the parameters α and ρ are tightly coupled to one another, since $\mathrm{var}\{\theta_n^{(t)}|\theta_n^{(t-1)}\} = \alpha^2\rho$. Consequently, if the initial choices of α and ρ

% Define key quantities obtained from AMP-MMV at iteration k:

$$\mathrm{E}[s_n|\bar{\boldsymbol{y}}] = \frac{\lambda_n \prod_{t=1}^{T} \overrightarrow{\pi}_n^{(t)}}{\lambda_n \prod_{t=1}^{T} \overrightarrow{\pi}_n^{(t)} + (1-\lambda_n) \prod_{t=1}^{T} (1-\overrightarrow{\pi}_n^{(t)})} \tag{Q1}$$

$$\tilde{v}_n^{(t)} \triangleq \mathrm{var}\{\theta_n^{(t)}|\bar{\boldsymbol{y}}\} = \left(\frac{1}{\overrightarrow{\kappa}_n^{(t)}} + \frac{1}{\overrightarrow{\psi}_n^{(t)}} + \frac{1}{\overleftarrow{\kappa}_n^{(t)}} \right)^{-1} \tag{Q2}$$

$$\tilde{\mu}_n^{(t)} \triangleq \mathrm{E}[\theta_n^{(t)}|\bar{\boldsymbol{y}}] = \tilde{v}_n^{(t)} \cdot \left(\frac{\overrightarrow{\eta}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}} + \frac{\overrightarrow{\xi}_n^{(t)}}{\overrightarrow{\psi}_n^{(t)}} + \frac{\overleftarrow{\eta}_n^{(t)}}{\overleftarrow{\kappa}_n^{(t)}} \right) \tag{Q3}$$

$v_n^{(t)} \triangleq \mathrm{var}\{x_n^{(t)}|\bar{\boldsymbol{y}}\}$ % See (A19) of Table 2.2

$\mu_n^{(t)} \triangleq \mathrm{E}[x_n^{(t)}|\bar{\boldsymbol{y}}]$ % See (A18) of Table 2.2

% EM update equations:

$$\lambda^{k+1} = \frac{1}{N} \sum_{n=1}^{N} \mathrm{E}[s_n|\bar{\boldsymbol{y}}] \tag{E1}$$

$$\zeta^{k+1} = \left(\frac{N(T-1)}{\rho^k} + \frac{N}{(\sigma^2)^k} \right)^{-1} \left(\frac{1}{(\sigma^2)^k} \sum_{n=1}^{N} \tilde{\mu}_n^{(1)} + \sum_{t=2}^{T} \sum_{n=1}^{N} \frac{1}{\alpha^k \rho^k} \left(\tilde{\mu}_n^{(t)} - (1-\alpha^k) \tilde{\mu}_n^{(t-1)} \right) \right) \tag{E2}$$

$$\alpha^{k+1} = \frac{1}{4N(T-1)} \left(\mathfrak{b} - \sqrt{\mathfrak{b}^2 + 8N(T-1)\mathfrak{c}} \right) \tag{E3}$$

where:

$$\mathfrak{b} \triangleq \frac{2}{\rho^k} \sum_{t=2}^{T} \sum_{n=1}^{N} \mathfrak{Re}\{\mathrm{E}[\theta_n^{(t)*} \theta_n^{(t-1)}|\bar{\boldsymbol{y}}]\} - \mathfrak{Re}\{(\tilde{\mu}_n^{(t)} - \tilde{\mu}_n^{(t-1)})^* \zeta^k\} - \tilde{v}_n^{(t-1)} - |\tilde{\mu}_n^{(t-1)}|^2$$

$$\mathfrak{c} \triangleq \frac{2}{\rho^k} \sum_{t=2}^{T} \sum_{n=1}^{N} \tilde{v}_n^{(t)} + |\tilde{\mu}_n^{(t)}|^2 + \tilde{v}_n^{(t-1)} + |\tilde{\mu}_n^{(t-1)}|^2 - 2\mathfrak{Re}\{\mathrm{E}[\theta_n^{(t)*} \theta_n^{(t-1)}|\bar{\boldsymbol{y}}]\}$$

$$\rho^{k+1} = \frac{1}{(\alpha^k)^2 N(T-1)} \sum_{t=2}^{T} \sum_{n=1}^{N} \tilde{v}_n^{(t)} + |\tilde{\mu}_n^{(t)}|^2 + (\alpha^k)^2 |\zeta^k|^2 - 2(1-\alpha^k) \mathfrak{Re}\{\mathrm{E}[\theta_n^{(t)*} \theta_n^{(t-1)}|\bar{\boldsymbol{y}}]\} - 2\alpha^k \mathfrak{Re}\{\tilde{\mu}_n^{(t)*} \zeta^k\} + 2\alpha^k (1-\alpha^k) \mathfrak{Re}\{\tilde{\mu}_n^{(t-1)*} \zeta^k\} + (1-\alpha^k)(\tilde{v}_n^{(t-1)} + |\tilde{\mu}_n^{(t-1)}|^2) \tag{E4}$$

$$\sigma_e^{2\ k+1} = \frac{1}{TM} \left(\sum_{t=1}^{T} \|\boldsymbol{y}^{(t)} - \boldsymbol{A}\boldsymbol{\mu}^{(t)}\|^2 + \mathbf{1}_N^T \boldsymbol{v}^{(t)} \right) \tag{E5}$$

Table 2.4: EM algorithm update equations for the signal model parameters of Section 2.2.

are too small, it is possible that the EM procedure will overcompensate on the first iteration by producing revised estimates of both parameters that are too large. This leads to an oscillatory behavior in the EM updates that can be effectively combated by avoiding updating both α and ρ on the same iteration.

2.6 Numerical Study

In this section we describe the results of an extensive numerical study that was conducted to explore the performance characteristics and tradeoffs of AMP-MMV. MATLAB code[7] was written to implement both the parallel and serial message schedules of Section 2.4.1, along with the EM parameter estimation procedure of Section 2.5.

For comparison to AMP-MMV, we tested two other Bayesian algorithms for the MMV problem, MSBL [13] and T-MSBL[8] [14], which have been shown to offer "best in class" performance on the MMV problem. We also included a recently proposed greedy algorithm designed specifically for highly correlated signals, subspace-augmented MUSIC[9] (SA-MUSIC), which has been shown to outperform MMV basis pursuit and several correlation-agnostic greedy methods [43]. Finally, we implemented the support-aware Kalman smoother (SKS), which, as noted in Section 2.3, provides a lower bound on the achievable MSE of any algorithm. To implement the SKS, we took advantage of the fact that $\bar{\boldsymbol{y}}$, $\bar{\boldsymbol{x}}$, and $\bar{\boldsymbol{\theta}}$ are jointly Gaussian when conditioned on the support, $\boldsymbol{s}$, and thus Fig. 2.1 becomes a Gaussian graphical model. Consequently, the sum-product algorithm yields closed-form expressions (i.e., no approximations are required) for each of the messages traversing the graph. Therefore, it is possible to obtain the desired posterior means (i.e., MMSE estimates of $\bar{\boldsymbol{x}}$) despite the fact that the graph is loopy [68, Claim 5].

In all of our experiments, performance was analyzed on synthetically generated datasets, and averaged over 250 independent trials. Since MSBL and T-MSBL were derived for real-valued signals, we used a real-valued equivalent of the signal model

[7] Code available at `ece.osu.edu/~schniter/turboAMPmmv`.

[8] Code available at `dsp.ucsd.edu/~zhilin/Software.html`.

[9] Code obtained through personal correspondence with authors.

described in Section 2.2, and ran a real-valued version of AMP-MMV. Our data generation procedure closely mirrors the one used to characterize T-MSBL in [14]. Unless otherwise stated, the measurement matrices were i.i.d. Gaussian random matrices with unit-norm columns, $T = 4$ measurement vectors were generated, the stationary variance of the amplitude process was set at $\sigma^2 \triangleq \frac{\alpha\rho}{2-\alpha} = 1$, and the noise variance σ_e^2 was set to yield an SNR of 25 dB.

Three performance metrics were considered throughout our tests. The first metric, which we refer to as the time-averaged normalized MSE (TNMSE), is defined as

$$\text{TNMSE}(\bar{\boldsymbol{x}}, \hat{\bar{\boldsymbol{x}}}) \triangleq \frac{1}{T}\sum_{t=1}^{T} \frac{\|\boldsymbol{x}^{(t)} - \hat{\boldsymbol{x}}^{(t)}\|_2^2}{\|\boldsymbol{x}^{(t)}\|_2^2},$$

where $\hat{\boldsymbol{x}}^{(t)}$ is an estimate of $\boldsymbol{x}^{(t)}$. The second metric, intended to gauge the accuracy of the recovered support, is the normalized support error rate (NSER), which is defined as the number of indices in which the true and estimated support differ, normalized by the cardinality of the true support $\mathcal{S}$. The third and final metric is runtime, which is an important metric given the prevalence of high-dimensional datasets.

The algorithms were configured and executed as follows: to obtain support estimates for MSBL, T-MSBL, and AMP-MMV, we adopted the technique utilized in [14] of identifying the K amplitude trajectories with the largest ℓ_2 norms as the support set, where $K \triangleq |\mathcal{S}|$. Note that this is an optimistic means of identifying the support, as it assumes that an oracle provides the true value of K. For this reason, we implemented an additional *non*-oracle-aided support estimate for AMP-MMV that consisted of those indices n for which $\hat{p}(s_n|\bar{\boldsymbol{y}}) > \frac{1}{2}$. In all simulations, AMP-MMV was given imperfect knowledge of the signal model parameters, and refined the initial parameter choices according to the EM update procedure given in Table 3.3. In

particular, the noise variance was initialized at $\sigma_e^2 = 1 \times 10^{-3}$. The remaining parameters were initialized agnostically using simple heuristics that made use of sample statistics derived from the available measurements, $\bar{\boldsymbol{y}}$. Equation (A22) of Table 2.2 was used to produce $\hat{\boldsymbol{x}}^{(t)}$, which corresponds to an MMSE estimate of $\boldsymbol{x}^{(t)}$ under AMP-MMV's estimated posteriors $\hat{p}(x_n^{(t)}|\bar{\boldsymbol{y}})$. In the course of running simulations, we monitored the residual energy, $\sum_{t=1}^{T} \|\boldsymbol{y}^{(t)} - \boldsymbol{A}\hat{\boldsymbol{x}}^{(t)}\|_2^2$, and would automatically switch the schedule, e.g., from parallel to serial, and/or change the maximum number of iterations whenever the residual energy exceeded a noise variance-dependent threshold. The SKS was given perfect parameter and support knowledge and was run until convergence. Both MSBL and T-MSBL were tuned in a manner recommended by the codes' authors. SA-MUSIC was given the true value of K, and upon generating an estimate of the support, $\hat{\mathcal{S}}$, a conditional MMSE signal estimate was produced, e.g., $\hat{\boldsymbol{x}}^{(t)} = \mathrm{E}[\boldsymbol{x}^{(t)}|\hat{\mathcal{S}}, \boldsymbol{y}^{(t)}]$.

2.6.1 Performance Versus Sparsity, M/K

As a first experiment, we studied how performance changes as a function of the measurements-to-active-coefficients ratio, M/K. For this experiment, $N = 5000$, $M = 1563$, and $T = 4$. The activity probability, λ, was swept over the range $[0.096, 0.22]$, implying that the ratio of measurements-to-active-coefficients, M/K, ranged from 1.42 to 3.26.

In Fig. 2.3, we plot the performance when the temporal correlation of the amplitudes is $1 - \alpha = 0.90$. For AMP-MMV, two traces appear on the NSER plot, with the $\bigcirc$ marker corresponding to the K-largest-trajectory-norm method of support estimation, and the $\triangle$ marker corresponding to the support estimate obtained from the posteriors $\hat{p}(s_n|\bar{\boldsymbol{y}})$. We see that, when $M/K \geq 2$, the TNMSE performance of both AMP-MMV and T-MSBL is almost identical to that of the oracle-aided SKS.

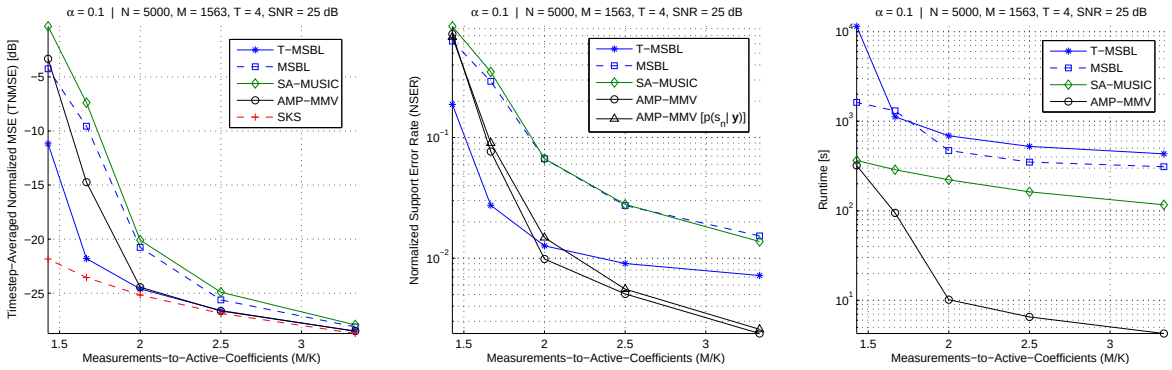


Figure 2.3: A plot of the TNMSE (in dB), NSER, and runtime of T-MSBL, MSBL, SA-MUSIC, AMP-MMV, and the SKS versus M/K. Correlation coefficient $1 - \alpha = 0.90$.

However, when $M/K < 2$, every algorithm's support estimation performance (NSER) degrades, and the TNMSE consequently grows. Indeed, when $M/K < 1.50$, all of the algorithms perform poorly compared to the SKS, although T-MSBL performs the best of the four. We also note the superior NSER performance of AMP-MMV over much of the range, even when using $p(s_n|\bar{\boldsymbol{y}})$ to estimate $\mathcal{S}$ (and thus not requiring apriori knowledge of K). From the runtime plot we see the tremendous efficiency of AMP-MMV. Over the region in which AMP-MMV is performing well (and thus not cycling through multiple configurations in vain), we see that AMP-MMV's runtime is more than one order-of-magnitude faster than SA-MUSIC, and two orders-of-magnitude faster than either T-MSBL or MSBL.

In Fig. 2.4 we repeat the same experiment, but with increased amplitude correlation $1 - \alpha = 0.99$. In this case we see that AMP-MMV and T-MSBL still offer a TNMSE performance that is comparable to the SKS when $M/K \geq 2.50$, whereas the performance of both MSBL and SA-MUSIC has degraded across-the-board. When $M/K < 2.5$, the NSER and TNMSE performance of AMP-MMV and T-MSBL decay

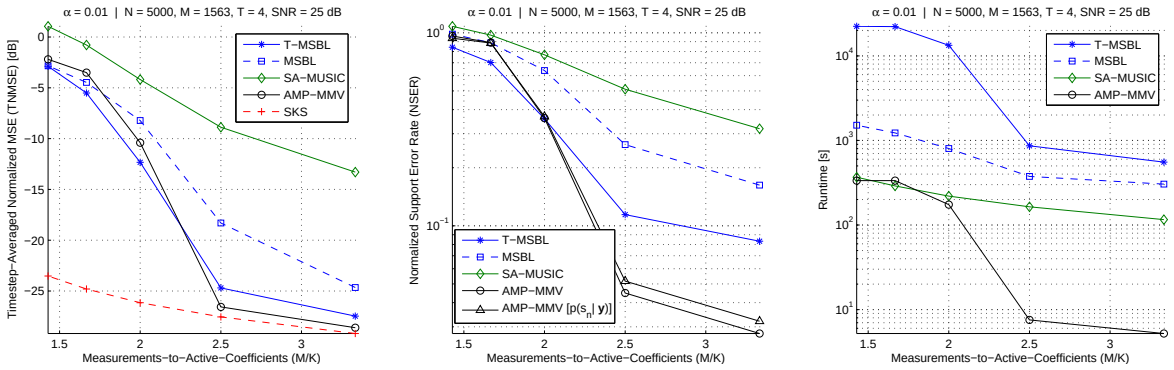


Figure 2.4: A plot of the TNMSE (in dB), NSER, and runtime of T-MSBL, MSBL, SA-MUSIC, AMP-MMV, and the SKS versus M/K. Correlation coefficient $1-\alpha = 0.99$.

sharply, and all the methods considered perform poorly compared to the SKS. Our finding that performance is adversely affected by increased temporal correlation is consistent with the theoretical and empirical findings of [13, 14, 35, 43]. Interestingly, the performance of the SKS shows a modest improvement compared to Fig. 2.3, reflecting the fact that the slower temporal variations of the amplitudes are easier to track when the support is known.

2.6.2 Performance Versus T

In a second experiment, we studied how performance is affected by the number of measurement vectors, T, used in the reconstruction. For this experiment, we used $N = 5000$, $M = N/5$, and $\lambda = 0.10$ ($M/K = 2$). Figure 2.5 shows the performance with a correlation of $1-\alpha = 0.90$. Comparing to Fig. 2.3, we see that MSBL's performance is strongly impacted by the reduced value of M. AMP-MMV and T-MSBL perform more-or-less equivalently across the range of T, although AMP-MMV does so with an order-of-magnitude reduction in complexity. It is interesting to

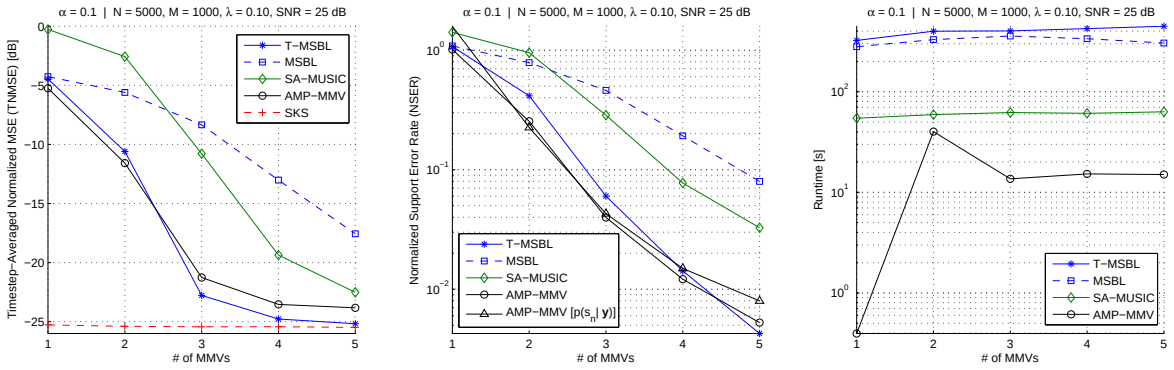


Figure 2.5: A plot of the TNMSE (in dB), NSER, and runtime of T-MSBL, MSBL, SA-MUSIC, AMP-MMV, and the SKS versus T. Correlation coefficient 1 - $\alpha = 0.90$.

observe that, in this problem regime, the SKS TNMSE bound is insensitive to the number of measurement vectors acquired.

2.6.3 Performance Versus SNR

To understand how AMP-MMV performs in low SNR environments, we conducted a test in which SNR was swept from 5 dB to 25 dB.[10] The problem dimensions were fixed at $N = 5000$, $M = N/5$, and $T = 4$. The sparsity rate, λ, was chosen to yield $M/K = 3$ measurements-per-active-coefficient, and the correlation was set at $1 - \alpha = 0.95$.

Our findings are presented in Fig. 2.6. Both T-MSBL and MSBL operate within 5 - 10 dB of the SKS in TNMSE across the range of SNRs, while AMP-MMV operates ≈ 5 dB from the SKS when the SNR is at or below 10 dB, and approaches the SKS

[10] In lower SNR regimes, learning rules for the noise variance are known to become less reliable [13, 14]. Still, for high-dimensional problems, a sub-optimal learning rule may be preferable to a computationally costly cross-validation procedure. For this reason, we ran all three Bayesian algorithms with a learning rule for the noise variance enabled.

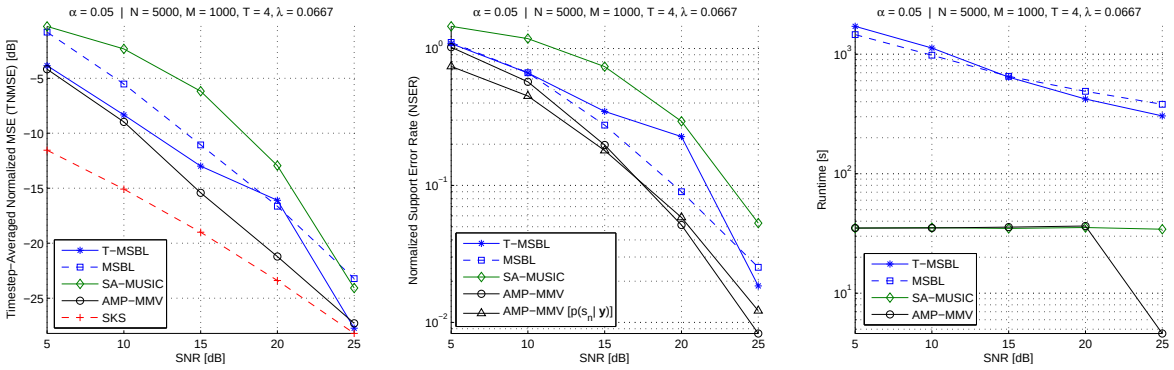


Figure 2.6: A plot of the TNMSE (in dB), NSER, and runtime of T-MSBL, MSBL, SA-MUSIC, AMP-MMV, and the SKS versus SNR. Correlation coefficient $1 - \alpha = 0.95$.

in performance as the SNR elevates. We also note that using AMP-MMV's posteriors on s_n to estimate the support does not appear to perform much worse than the K-largest-trajectory-norm method for high SNRs, and shows a slight advantage at low SNRs. The increase in runtime exhibited by AMP-MMV in this experiment is a consequence of our decision to configure AMP-MMV identically for all experiments; our initialization of the noise variance, σ_e^2, was more than an order-of-magnitude off over the majority of the SNR range, and thus AMP-MMV cycled through many different schedules in an effort to obtain an (unrealistic) residual energy. Runtime could be drastically improved in this experiment by using a more appropriate initialization of σ_e^2.

2.6.4 Performance Versus Undersampling Rate, N/M

As mentioned in Section 2.1, one of the principal aims of CS is to reduce the number of measurements that must be acquired while still obtaining a good solution. In the MMV problem, dramatic reductions in the sampling rate are possible. To illustrate this, in Fig. 2.7 we present the results of an experiment in which the undersampling

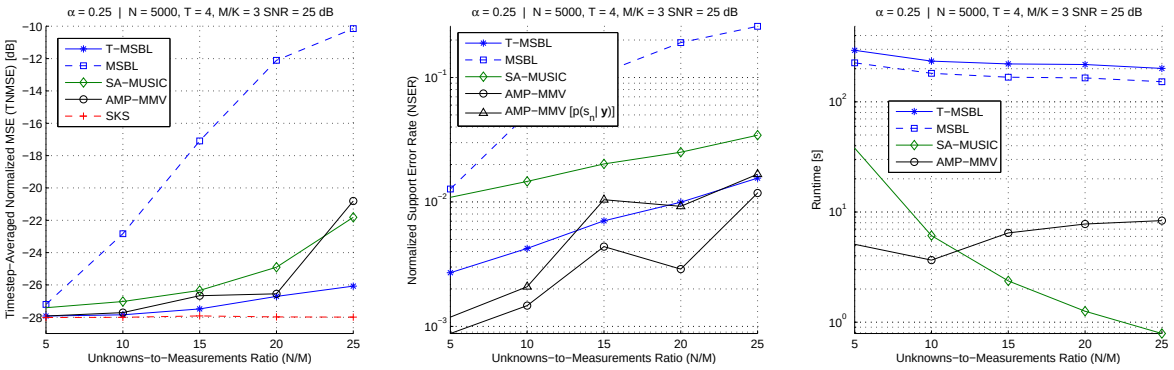


Figure 2.7: A plot of the TNMSE (in dB), NSER, and runtime of T-MSBL, MSBL, SA-MUSIC, AMP-MMV, and the SKS versus undersampling rate, N/M. Correlation coefficient $1 - \alpha = 0.75$.

factor, N/M, was varied from 5 to 25 unknowns-per-measurement. Specifically, N was fixed at 5000, while M was varied. λ was likewise adjusted in order to keep M/K fixed at 3 measurements-per-active-coefficient. In Fig. 2.7, we see that MSBL quickly departs from the SKS performance bound, whereas AMP-MMV, T-MSBL, and SA-MUSIC are able to remain close to the bound when $N/M \leq 20$. At $N/M = 25$, both AMP-MMV and SA-MUSIC have diverged from the bound, and, while still offering an impressive TNMSE, they are outperformed by T-MSBL. In conducting this test, we observed that AMP-MMV's performance is strongly tied to the number of smoothing iterations performed. Whereas for other tests, 5 smoothing iterations were often sufficient, in scenarios with a high degree of undersampling, (e.g., $N/M \geq 15$), $50 - 100$ smoothing iterations were often required to obtain good signal estimates. This suggests that messages must be exchanged between neighboring timesteps over many iterations in order to arrive at consensus in severely underdetermined problems.

2.6.5 Performance Versus Signal Dimension, N

As we have indicated throughout this paper, a key consideration of our method was ensuring that it would be suitable for high-dimensional problems. Our complexity analysis indicated that a single iteration of AMP-MMV could be completed in $\mathcal{O}(TNM)$ flops. This linear scaling of the complexity with respect to problem dimensions gives encouragement that our algorithm should efficiently handle large problems, but if the number of iterations required to obtain a solution grows too rapidly with problem size, our technique would be of limited practical utility. To ensure that this was not the case, we performed an experiment in which the signal dimension, N, was swept logarithmically over the range $[100, 10000]$. M was scaled proportionally such that $N/M = 3$. The sparsity rate was fixed at $\lambda = 0.15$ so that $M/K \approx 2$, and the correlation was set at $1 - \alpha = 0.95$.

The results of this experiment are provided in Fig. 2.8. Several features of these plots are of interest. First, we observe that the performance of every algorithm improves noticeably as problem dimensions grow from $N = 100$ to $N = 1000$, with AMP-MMV and T-MSBL converging in TNMSE performance to the SKS bound. The second observation that we point out is that AMP-MMV works extremely quickly. Indeed, a problem with $NT = 40000$ unknowns can be solved accurately in just under 30 seconds. Finally, we note that at small problem dimensions, AMP-MMV is not as quick as either MSBL or SA-MUSIC, however AMP-MMV scales with increasing problem dimensions more favorably than the other methods; at $N = 10000$ we note that AMP-MMV runs at least two orders-of-magnitude faster than the other techniques.

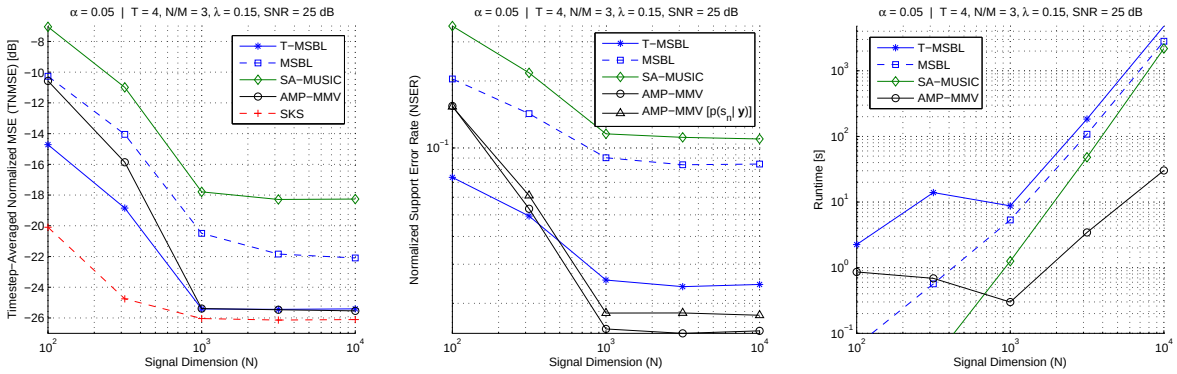


Figure 2.8: A plot of the TNMSE (in dB), NSER, and runtime of T-MSBL, MSBL, SA-MUSIC, AMP-MMV, and the SKS versus signal dimension, N. Correlation coefficient $1 - \alpha = 0.95$.

2.6.6 Performance With Time-Varying Measurement Matrices

In all of the previous experiments, we considered the standard MMV problem (2.1), in which all of the measurement vectors were acquired using a single, common measurement matrix. While this setup is appropriate for many tasks, there are a number of practical applications in which a joint-sparse signal is measured through distinct measurement matrices.

To better understand what, if any, gains can be obtained from diversity in the measurement matrices, we designed an experiment that explored how performance is affected by the rate-of-change of the measurement matrix over time. For simplicity, we considered a first-order Gauss-Markov random process to describe how a given measurement matrix changed over time. Specifically, we started with a matrix whose columns were drawn i.i.d. Gaussian as in previous experiments, which was then used as the measurement matrix to collect the measurements at timestep $t = 1$. At

subsequent timesteps, the matrix evolved according to

$$\boldsymbol{A}^{(t)} = (1-\beta)\boldsymbol{A}^{(t-1)} + \beta\boldsymbol{U}^{(t)}, \tag{2.15}$$

where $\boldsymbol{U}^{(t)}$ was a matrix whose elements were drawn i.i.d. Gaussian, with a variance chosen such that the column norm of $\boldsymbol{A}^{(t)}$ would (in expectation) equal one.

In the test, β was swept over a range, providing a quantitative measure of the rate-of-change of the measurement matrix over time. Clearly, $\beta = 0$ would correspond to the standard MMV problem, while $\beta = 1$ would represent a collection of statistically independent measurement matrices.

In Fig. 2.9 we show the performance when $N = 5000$, $N/M = 30$, $M/K = 2$, and the correlation is $1-\alpha = 0.99$. For the standard MMV problem, this configuration is effectively impossible. Indeed, for $\beta < 0.03$, we see that AMP-MMV is entirely failing at recovering the signal. However, once $\beta \approx 0.08$, we see that the NSER has dropped dramatically, as has the TNMSE. Once $\beta \geq 0.10$, AMP-MMV is performing almost to the level of the noise. As this experiment should hopefully convince the reader, even modest amounts of diversity in the measurement process can enable accurate reconstruction in operating environments that are otherwise impossible.

2.7 Conclusion

In this chapter we introduced AMP-MMV, a Bayesian message passing algorithm for solving the MMV problem (2.1) when temporal correlation is present in the amplitudes of the non-zero signal coefficients. Our algorithm, which leverages Donoho, Maleki, and Montanari's AMP framework [25], performs rapid inference on high-dimensional MMV datasets. In order to establish a reference point for the quality of solutions obtained by AMP-MMV, we described and implemented the oracle-aided

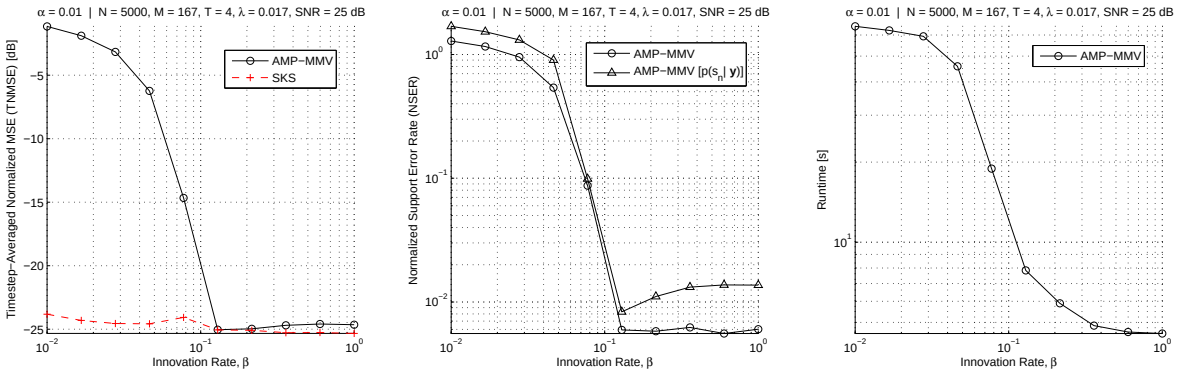


Figure 2.9: A plot of the TNMSE (in dB), NSER, and runtime of AMP-MMV and the SKS versus rate-of-change of the measurement matrix, β. Correlation coefficient $1 - \alpha = 0.99$.

support-aware Kalman smoother (SKS). In numerical experiments, we found a range of problems over which AMP-MMV performed nearly as well as the SKS, despite the fact that AMP-MMV was given crude hyperparameter initializations that were refined from the data using an expectation-maximization algorithm. In comparing against two alternative Bayesian techniques, and one greedy technique, we found that AMP-MMV offers an unrivaled performance-complexity tradeoff, particular in high-dimensional settings. We also demonstrated that substantial gains can be obtained in the MMV problem by incorporating diversity into the measurement process. Such diversity is particularly important in settings where the temporal correlation between coefficient amplitudes is substantial.

CHAPTER 3

THE DYNAMIC COMPRESSIVE SENSING PROBLEM

"On the contrary, Watson, you can see everything. You fail, however, to reason from what you see. You are too timid in drawing your inferences."

- Sherlock Holmes

In Chapter 2, we studied the MMV CS problem of recovering a temporally correlated, sparse time series that possessed a common support. In this chapter,[1] we consider a generalization of the MMV CS problem known as the *dynamic compressive sensing* (dynamic CS) problem, in which the sparse time series has a slowly time-varying, rather than time-invariant, support. Such a problem finds application in, e.g., dynamic MRI [45], high-speed video capture [69], and underwater channel estimation [9].

3.1 Introduction

Framed mathematically, the objective of the dynamic CS problem is to recover the time series $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}\}$, where $\boldsymbol{x}^{(t)} \in \mathbb{C}^N$ is the signal at timestep t, from a time

[1] Work presented in this chapter is largely excerpted from a manuscript co-authored with Philip Schniter, entitled "Dynamic Compressive Sensing of Time-Varying Signals via Approximate Message Passing." [37]

series of measurements, $\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(T)}\}$. Each $\boldsymbol{y}^{(t)} \in \mathbb{C}^M$ is obtained from the linear measurement process,

$$\boldsymbol{y}^{(t)} = \boldsymbol{A}^{(t)}\boldsymbol{x}^{(t)} + \boldsymbol{e}^{(t)}, \qquad t = 1, \ldots, T, \tag{3.1}$$

with $\boldsymbol{e}^{(t)}$ representing corrupting noise. The measurement matrix $\boldsymbol{A}^{(t)}$ (which may be time-varying or time-invariant, i.e., $\boldsymbol{A}^{(t)} = \boldsymbol{A} \; \forall \; t$) is known in advance, and is generally wide, leading to an underdetermined system of equations. The problem is regularized by assuming that $\boldsymbol{x}^{(t)}$ is sparse (or compressible),[2] having relatively few non-zero (or large) entries.

In many real-world scenarios, the underlying time-varying sparse signal exhibits substantial temporal correlation. This temporal correlation may manifest itself in two interrelated ways: *(i)* the support of the signal may change slowly over time [45, 46, 69–72], and *(ii)* the amplitudes of the large coefficients may vary smoothly in time.

In such scenarios, incorporating an appropriate model of temporal structure into a recovery technique makes it possible to drastically outperform structure-agnostic CS algorithms. From an analytical standpoint, Vaswani and Lu demonstrate that the restricted isometry property (RIP) sufficient conditions for perfect recovery in the dynamic CS problem are significantly weaker than those found in the traditional single measurement vector (SMV) CS problem when accounting for the additional structure [48]. In this chapter, we take a Bayesian approach to modeling this structure, which contrasts those dynamic CS algorithms inspired by convex relaxation, such as

[2] Without loss of generality, we assume $\boldsymbol{x}^{(t)}$ is sparse/compressible in the canonical basis. Other sparsifying bases can be incorporated into the measurement matrix $\boldsymbol{A}^{(t)}$ without changing our model.

the Dynamic LASSO [46] and the Modified-CS algorithm [48]. Our Bayesian framework is also distinct from those hybrid techniques that blend elements of Bayesian dynamical models like the Kalman filter with more traditional CS approaches of exploiting sparsity through convex relaxation [45, 47] or greedy methods [73].

In particular, we propose a probabilistic model that treats the time-varying signal support as a set of independent binary Markov processes and the time-varying coefficient amplitudes as a set of independent Gauss-Markov processes. As detailed in Section 3.2, this model leads to coefficient marginal distributions that are Bernoulli-Gaussian (i.e., "spike-and-slab"). Later, in Section 3.5, we describe a generalization of the aforementioned model that yields Bernoulli-Gaussian-mixture coefficient marginals with an arbitrary number of mixture components. The models that we propose thus differ substantially from those used in other Bayesian approaches to dynamic CS, [20] and [18]. In particular, Sejdinović et al. [20] combine a linear Gaussian dynamical system model with a sparsity-promoting Gaussian-scale-mixture prior, while Shahrasbi et al. [18] employ a particular spike-and-slab Markov model that couples amplitude evolution together with support evolution.

Our inference method also differs from those used in the alternative Bayesian dynamic CS algorithms [20] and [18]. In [20], Sejdinović et al. perform inference via a sequential Monte Carlo sampler [74]. Sequential Monte Carlo techniques are appealing for their applicability to complicated non-linear, non-Gaussian inference tasks like the Bayesian dynamic CS problem. Nevertheless, there are a number of important practical issues related to selection of the importance distribution, choice of the resampling method, and the number of sample points to track, since in principle one must increase the number of points exponentially over time to combat degeneracy [74]. Additionally, Monte Carlo techniques can be computationally expensive in high-dimensional inference problems. An alternative inference procedure that has

recently proven successful in a number of applications is loopy belief propagation (LBP) [28]. In [18], Shahrasbi et al. extend the conventional LBP method proposed in [59] for standard CS under a sparse measurement matrix $\boldsymbol{A}$ to the case of dynamic CS under sparse $\boldsymbol{A}^{(t)}$. Nevertheless, the confinement to sparse measurement matrices is very restrictive, and, without this restriction, the methods of [18, 59] become computationally intractable.

Our inference procedure is based on the recently proposed framework of approximate message passing (AMP) [25], and in particular its "turbo" extension [53]. AMP, an unconventional form of LBP, was originally proposed for standard CS with a dense measurement matrix [25], and its noteworthy properties include: *(i)* a rigorous analysis (as $M, N \to \infty$ with M/N fixed, under i.i.d. sub-Gaussian $\boldsymbol{A}$) establishing that its solutions are governed by a state-evolution whose fixed points are optimal in several respects [29], and *(ii)* extremely fast runtimes (as a consequence of the fact that it needs relatively few iterations, each requiring only one multiplication by $\boldsymbol{A}$ and its transpose). The turbo-AMP framework originally proposed in [53] offers a way to extend AMP to structured-sparsity problems such as compressive imaging [16], joint communication channel/symbol estimation [15], and—as we shall see in this chapter—the dynamic CS problem.

Our work makes several contributions to the existing literature on dynamic CS. First and foremost, the DCS-AMP algorithm that we develop offers an unrivaled combination of speed (e.g., its computational complexity grows only linearly in the problem dimensions M, N, and T) and reconstruction accuracy, as we demonstrate on both synthetic and real-world signals. Ours is the first work to exploit the speed and accuracy of loopy belief propagation (and, in particular, AMP) in the dynamic CS setting, accomplished by embedding AMP within a larger Bayesian inference algorithm. Second, we propose an expectation-maximization [65] procedure to automatically

learn the parameters of our statistical model, as described in Section 3.4, avoiding a potentially complicated "tuning" problem. The ability to automatically calibrate algorithm parameters is especially important when working with real-world data, but is not provided by many of the existing dynamic CS algorithms (e.g., [20,45–48,73]). In addition, our learned model parameters provide a convenient and interpretable characterization of time-varying signals in a way that, e.g., Lagrange multipliers do not. Third, DCS-AMP provides a unified means of performing both filtering, where estimates are obtained sequentially using only past observations, and smoothing, where each estimate enjoys the knowledge of past, current, and future observations. In contrast, the existing dynamic CS schemes can support either filtering, or smoothing, but not both.

The notation used in the remainder of this chapter adheres to the convention established in Section 2.1.1.

3.2 Signal Model

We assume that the measurement process can be accurately described by the linear model of (3.1). We further assume that $\boldsymbol{A}^{(t)} \in \mathbb{C}^{M \times N}$, $t = 1, \ldots, T$, are measurement matrices known in advance, whose columns have been scaled to be of unit norm.[3] We model the noise as a stationary, circularly symmetric, additive white Gaussian noise (AWGN) process, with $\boldsymbol{e}^{(t)} \sim \mathcal{CN}(\boldsymbol{0}, \sigma_e^2 \boldsymbol{I}_M) \; \forall \; t$.

As noted in Section 3.1, the sparse time series, $\{\boldsymbol{x}^{(t)}\}_{t=1}^{T}$, often exhibits a high degree of correlation from one timestep to the next. In what follows, we model this correlation through a slow time-variation of the signal support, and a smooth evolution of the amplitudes of the non-zero coefficients. To do so, we introduce two

[3] Our algorithm can be generalized to support $\boldsymbol{A}^{(t)}$ without equal-norm columns, a time-varying number of measurements, $M^{(t)}$, and real-valued matrices/signals as well.

hidden random processes, $\{\boldsymbol{s}^{(t)}\}_{t=1}^{T}$ and $\{\boldsymbol{\theta}^{(t)}\}_{t=1}^{T}$. The binary vector $\boldsymbol{s}^{(t)} \in \{0,1\}^N$ describes the support of $\boldsymbol{x}^{(t)}$, denoted $\mathcal{S}^{(t)}$, while the vector $\boldsymbol{\theta}^{(t)} \in \mathbb{C}^N$ describes the amplitudes of the active elements of $\boldsymbol{x}^{(t)}$. Together, $\boldsymbol{s}^{(t)}$ and $\boldsymbol{\theta}^{(t)}$ completely characterize $\boldsymbol{x}^{(t)}$ as follows:

$$x_n^{(t)} = s_n^{(t)} \cdot \theta_n^{(t)} \quad \forall n, t. \tag{3.2}$$

Therefore, $s_n^{(t)} = 0$ sets $x_n^{(t)} = 0$ and $n \notin \mathcal{S}^{(t)}$, while $s_n^{(t)} = 1$ sets $x_n^{(t)} = \theta_n^{(t)}$ and $n \in \mathcal{S}^{(t)}$.

To model slow changes in the support $\mathcal{S}^{(t)}$ over time, we model the n^{th} coefficient's support across time, $\{s_n^{(t)}\}_{t=1}^{T}$, as a Markov chain defined by two transition probabilities: $p_{10} \triangleq \Pr\{s_n^{(t)}=1|s_n^{(t-1)}=0\}$, and $p_{01} \triangleq \Pr\{s_n^{(t)} = 0|s_n^{(t-1)} = 1\}$, and employ independent chains across $n = 1, \ldots, N$. We further assume that each Markov chain operates in steady-state, such that $\Pr\{s_n^{(t)} = 1\} = \lambda \; \forall \, n, t$. This steady-state assumption implies that these Markov chains are completely specified by the parameters λ and p_{01}, which together determine the remaining transition probability $p_{10} = \lambda p_{01}/(1-\lambda)$. Depending on how p_{01} is chosen, the prior distribution can favor signals that exhibit a nearly static support across time, or it can allow for signal supports that change substantially from timestep to timestep. For example, it can be shown that $1/p_{01}$ specifies the average run length of a sequence of ones in the Markov chains.

The second form of temporal structure that we capture in our signal model is the correlation in active coefficient amplitudes across time. We model this correlation through independent stationary steady-state Gauss-Markov processes for each n, wherein $\{\theta_n^{(t)}\}_{t=1}^{T}$ evolves in time according to

$$\theta_n^{(t)} = (1-\alpha)\big(\theta_n^{(t-1)} - \zeta\ \big) + \alpha w_n^{(t)} + \zeta, \tag{3.3}$$

where $\zeta \in \mathbb{C}$ is the mean of the process, $w_n^{(t)} \sim \mathcal{CN}(0, \rho)$ is an i.i.d. circular white Gaussian perturbation, and $\alpha \in [0, 1]$ controls the temporal correlation. At one extreme, $\alpha = 0$, the amplitudes are totally correlated, (i.e., $\theta_n^{(t)} = \theta_n^{(t-1)}$), while at the other extreme, $\alpha = 1$, the amplitudes evolve according to an uncorrelated Gaussian random process with mean ζ.

At this point, we would like to make a few remarks about our signal model. First, due to (3.2), it is clear that $p\big(x_n^{(t)}|s_n^{(t)}, \theta_n^{(t)}\big) = \delta\big(x_n^{(t)} - s_n^{(t)}\theta_n^{(t)}\big)$, where $\delta(\cdot)$ is the Dirac delta function. By marginalizing out $s_n^{(t)}$ and $\theta_n^{(t)}$, one finds that

$$p(x_n^{(t)}) = (1 - \lambda)\delta(x_n^{(t)}) + \lambda\, \mathcal{CN}(x_n^{(t)}; \zeta, \sigma^2), \tag{3.4}$$

where $\sigma^2 \triangleq \frac{\alpha\rho}{2-\alpha}$ is the steady-state variance of $\theta_n^{(t)}$. Equation (3.4) is a Bernoulli-Gaussian or "spike-and-slab" distribution, which is an effective sparsity-promoting prior due to the point-mass at $x_n^{(t)} = 0$. Second, we observe that the amplitude random process, $\{\boldsymbol{\theta}^{(t)}\}_{t=1}^T$, evolves independently from the sparsity pattern random process, $\{\boldsymbol{s}^{(t)}\}_{t=1}^T$. As a result of this modeling choice, there can be significant hidden amplitudes $\theta_n^{(t)}$ associated with inactive coefficients (those for which $s_n^{(t)} = 0$). Consequently, $\theta_n^{(t)}$ should be viewed as the amplitude of $x_n^{(t)}$ *conditioned* on $s_n^{(t)} = 1$. Lastly, we note that higher-order Markov processes and/or more complex coefficient marginals could be considered within the framework we propose, however, to keep development simple, we restrict our attention to first-order Markov processes and Bernoulli-Gaussian marginals until Section 3.5, where we describe an extension of the above signal model that yields Bernoulli-Gaussian-mixture marginals.

3.3 The DCS-AMP Algorithm

In this section we will describe the DCS-AMP algorithm, which efficiently and accurately estimates the marginal posterior distributions of $\{x_n^{(t)}\}$, $\{\theta_n^{(t)}\}$, and $\{s_n^{(t)}\}$ from the observed measurements $\{\boldsymbol{y}^{(t)}\}_{t=1}^T$, thus enabling both soft estimation and soft support detection. The use of soft support information is particularly advantageous, as it means that the algorithm need never make a firm (and possibly erroneous) decision about the support that can propagate errors across many timesteps. As mentioned in Section 3.1, DCS-AMP can perform either filtering or smoothing.

The algorithm we develop is designed to exploit the statistical structure inherent in our signal model. By defining $\bar{\boldsymbol{y}}$ to be the collection of all measurements, $\{\boldsymbol{y}^{(t)}\}_{t=1}^T$ (and defining $\bar{\boldsymbol{x}}$, $\bar{\boldsymbol{s}}$, and $\bar{\boldsymbol{\theta}}$ similarly), the posterior joint distribution of the signal, support, and amplitude time series, given the measurement time series, can be expressed using Bayes' rule as

$$p(\bar{\boldsymbol{x}}, \bar{\boldsymbol{s}}, \bar{\boldsymbol{\theta}} | \bar{\boldsymbol{y}}) \propto \prod_{t=1}^{T} \left(\prod_{m=1}^{M} p(y_m^{(t)} | \boldsymbol{x}^{(t)}) \prod_{n=1}^{N} p(x_n^{(t)} | s_n^{(t)}, \theta_n^{(t)}) p(s_n^{(t)} | s_n^{(t-1)}) p(\theta_n^{(t)} | \theta_n^{(t-1)}) \right), \tag{3.5}$$

where $\propto$ indicates proportionality up to a constant scale factor, $p(s_n^{(1)}|s_n^{(0)}) \triangleq p(s_n^{(1)})$, and $p(\theta_n^{(1)}|\theta_n^{(0)}) \triangleq p(\theta_n^{(1)})$. By inspecting (3.5), we see that the posterior joint distribution decomposes into the product of many distributions that only depend on small subsets of variables. A graphical representation of such decompositions is given by the *factor graph*, which is an undirected bipartite graph that connects the pdf "factors" of (3.5) with the random variables that constitute their arguments [55]. In Table 3.1, we introduce the notation that we will use for the factors of our signal model, showing the correspondence between the factor labels and the underlying distributions they represent, as well as the specific functional form assumed by each factor. The associated factor graph for the posterior joint distribution of (3.5) is

Factor	Distribution	Functional Form
$g_m^{(t)}\big(\boldsymbol{x}^{(t)}\big)$	$p\big(y_m^{(t)}\big\vert\boldsymbol{x}^{(t)}\big)$	$\mathcal{CN}\big(y_m^{(t)};\boldsymbol{a}_m^{(t)\,T}\boldsymbol{x}^{(t)},\sigma_e^2\big)$
$f_n^{(t)}\big(x_n^{(t)},s_n^{(t)},\theta_n^{(t)}\big)$	$p\big(x_n^{(t)}\big\vert s_n^{(t)},\theta_n^{(t)}\big)$	$\delta\big(x_n^{(t)}-s_n^{(t)}\theta_n^{(t)}\big)$
$h_n^{(1)}\big(s_n^{(1)}\big)$	$p\big(s_n^{(1)}\big)$	$\big(1-\lambda\big)^{1-s_n^{(1)}}\lambda^{s_n^{(1)}}$
$h_n^{(t)}\big(s_n^{(t)},s_n^{(t-1)}\big)$	$p\big(s_n^{(t)}\big\vert s_n^{(t-1)}\big)$	$\begin{cases}(1-p_{10})^{1-s_n^{(t)}}p_{10}{}^{s_n^{(t)}}, & s_n^{(t-1)}=0\\ p_{01}{}^{1-s_n^{(t)}}(1-p_{01})^{s_n^{(t)}}, & s_n^{(t-1)}=1\end{cases}$
$d_n^{(1)}\big(\theta_n^{(1)}\big)$	$p\big(\theta_n^{(1)}\big)$	$\mathcal{CN}\big(\theta_n^{(1)};\zeta,\sigma^2\big)$
$d_n^{(t)}\big(\theta_n^{(t)},\theta_n^{(t-1)}\big)$	$p\big(\theta_n^{(t)}\big\vert\theta_n^{(t-1)}\big)$	$\mathcal{CN}\big(\theta_n^{(t)};(1-\alpha)\theta_n^{(t-1)}+\alpha\zeta,\alpha^2\rho\big)$

Table 3.1: The factors, underlying distributions, and functional forms associated with our signal model

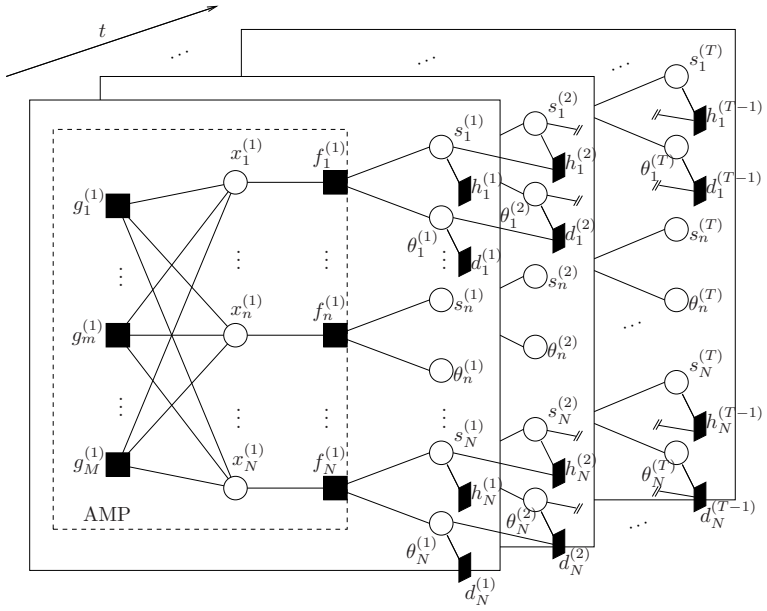


Figure 3.1: Factor graph representation of the joint posterior distribution of (3.5).

shown in Fig. 3.1, labeled according to Table 3.1. Filled squares represent factors, while circles represent random variables.

As seen in Fig. 3.1, all of the variables needed at a given timestep can be visualized as lying in a plane, with successive planes stacked one after another in time. We will

refer to these planes as "frames". The temporal correlation of the signal supports is illustrated by the $h_n^{(t)}$ factor nodes that connect the $s_n^{(t)}$ variable nodes between neighboring frames. Likewise, the temporal correlation of the signal amplitudes is expressed by the interconnection of $d_n^{(t)}$ factor nodes and $\theta_n^{(t)}$ variable nodes. For visual clarity, these factor nodes have been omitted from the middle portion of the factor graph, appearing only at indices $n = 1$ and $n = N$, but should in fact be present for all indices $n = 1, \ldots, N$. Since the measurements $\{y_m^{(t)}\}$ are observed variables, they have been incorporated into the $g_m^{(t)}$ factor nodes.

The algorithm that we develop can be viewed as an approximate implementation of belief propagation (BP) [56], a message passing algorithm for performing inference on factor graphs that describe probabilistic models. When the factor graph is cycle-free, belief propagation is equivalent to the more general sum-product algorithm [55], which is a means of computing the marginal functions that result from summing (or integrating) a multivariate function over all possible input arguments, with one argument held fixed, (i.e., marginalizing out all but one variable). In the context of BP, these marginal functions are the marginal distributions of random variables. Thus, given measurements $\bar{\boldsymbol{y}}$ and the factorization of the posterior joint distribution $p\big(\bar{\boldsymbol{x}}, \bar{\boldsymbol{s}}, \bar{\boldsymbol{\theta}} \big| \bar{\boldsymbol{y}}\big)$, DCS-AMP computes (approximate) posterior marginals of $x_n^{(t)}$, $s_n^{(t)}$, and $\theta_n^{(t)}$. In "filtering" mode, our algorithm would therefore return, e.g., $p\big(x_n^{(t)} \big| \{\boldsymbol{y}^{(t)}\}_{t=1}^{t}\big)$, while in "smoothing" mode it would return $p\big(x_n^{(t)} \big| \{\boldsymbol{y}^{(t)}\}_{t=1}^{T}\big)$. From these marginals, one can compute, e.g., minimum mean-squared error (MMSE) estimates. The factor graph of Fig. 3.1 contains many short cycles, however, and thus the convergence of loopy BP cannot be guaranteed [55].[4] Despite this, loopy BP has been shown to

[4]However, it is worth noting that in the past decade much work has been accomplished in identifying specific situations under which loopy BP *is* guaranteed to converge, e.g., [29, 68, 75–77].

perform extremely well in a number of different applications, including turbo decoding [78], computer vision [57], and compressive sensing [16, 19, 25, 26, 32, 53, 59].

3.3.1 Message scheduling

In loopy factor graphs, there are a number of ways to schedule, or sequence, the messages that are exchanged between nodes. The choice of a schedule can impact not only the rate of convergence of the algorithm, but also the likelihood of convergence as well [79]. We propose a schedule (an evolution of the "turbo" schedule proposed in [53]) for DCS-AMP that is straightforward to implement, suitable for both filtering and smoothing applications, and empirically yields quickly converging estimates under a variety of diverse operating conditions.

Our proposed schedule can be broken down into four distinct steps, which we will refer to using the mnemonics **(into)**, **(within)**, **(out)**, and **(across)**. At a particular timestep t, the **(into)** step involves passing messages that provide current beliefs about the state of the relevant support variables, $\{s_n^{(t)}\}_{n=1}^N$, and amplitude variables, $\{\theta_n^{(t)}\}_{n=1}^N$, laterally *into* the dashed AMP box within frame t. (Recall Fig. 3.1.) The **(within)** step makes use of these incoming messages, together with the observations available in that frame, $\{y_m^{(t)}\}_{m=1}^M$, to exchange messages *within* the dashed AMP box of frame t, thus generating estimates of the marginal posteriors of the signal variables $\{x_n^{(t)}\}_{n=1}^N$. Using these posterior estimates, the **(out)** step propagates messages *out* of the dashed AMP box, providing updated beliefs about the state of $\{s_n^{(t)}\}_{n=1}^N$ and $\{\theta_n^{(t)}\}_{n=1}^N$. Lastly, the **(across)** step involves transmitting messages *across* neighboring frames, using the updated beliefs about $\{s_n^{(t)}\}_{n=1}^N$ and $\{\theta_n^{(t)}\}_{n=1}^N$ to influence the beliefs about $\{s_n^{(t+1)}\}_{n=1}^N$ and $\{\theta_n^{(t+1)}\}_{n=1}^N$ $\big($or $\{s_n^{(t-1)}\}_{n=1}^N$ and $\{\theta_n^{(t-1)}\}_{n=1}^N\big)$.

The procedures for filtering and smoothing both start in the same way. At the initial $t = 1$ frame, steps **(into)**, **(within)** and **(out)** are performed in succession.

Next, step **(across)** is performed to pass messages from $\{s_n^{(1)}\}_{n=1}^N$ and $\{\theta_n^{(1)}\}_{n=1}^N$ to $\{s_n^{(2)}\}_{n=1}^N$ and $\{\theta_n^{(2)}\}_{n=1}^N$. Then at frame $t = 2$ the same set of steps are executed, concluding with messages propagating to $\{s_n^{(3)}\}_{n=1}^N$ and $\{\theta_n^{(3)}\}_{n=1}^N$. This process continues until steps **(into)**, **(within)** and **(out)** have been completed at the terminal frame, T. At this point, DCS-AMP has completed what we call a single forward pass. If the objective was to perform filtering, DCS-AMP terminates at this point, since only causal measurements have been used to estimate the marginal posteriors. If instead the objective is to obtain smoothed, non-causal estimates, then information begins to propagate backwards in time, i.e., step **(across)** moves messages from $\{s_n^{(T)}\}_{n=1}^N$ and $\{\theta_n^{(T)}\}_{n=1}^N$ to $\{s_n^{(T-1)}\}_{n=1}^N$ and $\{\theta_n^{(T-1)}\}_{n=1}^N$. Steps **(into)**, **(within)**, **(out)**, and **(across)** are performed at frame $T-1$, with messages bound for frame $T-2$. This continues until the initial frame is reached. At this point DCS-AMP has completed what we term as a single forward/backward pass. Multiple such passes, indexed by the variable k, can be carried out until a convergence criterion is met or a maximum number of passes has been performed.

3.3.2 Implementing the message passes

We now provide some additional details as to how the above four steps are implemented. To aid our discussion, in Fig. 3.2 we summarize the form of the messages that pass between the various factor graph nodes, focusing primarily on a single coefficient index n at an intermediate frame t. Directed edges indicate the direction that messages are moving. In the **(across)** phase, we only illustrate the messages involved in a forward pass for the amplitude variables, and leave out a graphic for the corresponding backward pass, as well as graphics for the support variable **(across)** phase. Note that, to be applicable at frame T, the factor node $d_n^{(t+1)}$ and its associated edge should be removed. The figure also introduces the notation that we adopt for

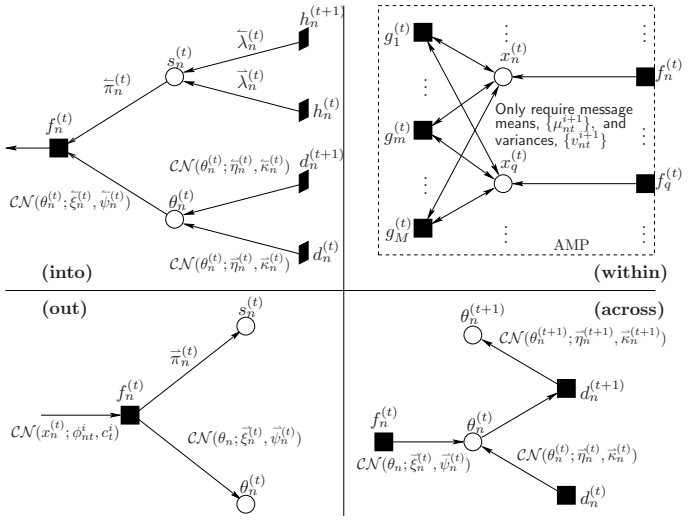


Figure 3.2: A summary of the four message passing phases, including message notation and form. See the pseudocode of Table 3.2 for the precise message update computations.

the different variables that serve to parameterize the messages. We use the notation $\nu_{a\to b}(\cdot)$ to denote a message passing from node a to a connected node b. For Bernoulli message pdfs, we show only the non-zero probability, e.g., $\overrightarrow{\lambda}_n^{(t)} = \nu_{h_n^{(t)}\to s_n^{(t)}}(s_n^{(t)} = 1)$.

To perform step **(into)**, the messages from the factors $h_n^{(t)}$ and $h_n^{(t+1)}$ to $s_n^{(t)}$ are used to set $\overleftarrow{\pi}_n^{(t)}$, the message from $s_n^{(t)}$ to $f_n^{(t)}$. Likewise, the messages from the factors $d_n^{(t)}$ and $d_n^{(t+1)}$ to $\theta_n^{(t)}$ are used to determine the message from $\theta_n^{(t)}$ to $f_n^{(t)}$. When performing filtering, or the first forward pass of smoothing, no meaningful information should be conveyed from the $h_n^{(t+1)}$ and $d_n^{(t+1)}$ factors. This can be accomplished by initializing $\left(\overleftarrow{\lambda}_n^{(t)}, \overleftarrow{\eta}_n^{(t)}, \overleftarrow{\kappa}_n^{(t)}\right)$ with the values $(\frac{1}{2}, 0, \infty)$.

In step **(within)**, messages must be exchanged between the $\left\{x_n^{(t)}\right\}_{n=1}^N$ and $\left\{g_m^{(t)}\right\}_{m=1}^M$ nodes. When $\boldsymbol{A}^{(t)}$ is not a sparse matrix, this will imply a dense network of connections between these nodes. Recall that in Section 2.4.2, we leveraged an AMP algorithm in the MMV problem to manage the computationally intensive message passes in the dense subgraph of Fig. 2.1 consisting of these nodes. Such an approach

is equally well-suited in the DCS problem. As in Chapter 2, the local prior for the signal model of Section 3.2 is a Bernoulli-Gaussian, namely

$$\nu_{f_n^{(t)}\to x_n^{(t)}}(x_n^{(t)}) = (1-\overleftarrow{\pi}_n^{(t)})\delta(x_n^{(t)}) + \overleftarrow{\pi}_n^{(t)}\mathcal{CN}(x_n^{(t)}; \overleftarrow{\xi}_n^{(t)}, \overleftarrow{\psi}_n^{(t)}).$$

The specific AMP updates for our model are given by (A17)-(A21) in Table 3.2.

Recall also from Section 2.4.2 that we needed to devise an approximation scheme to manage the $f_n^{(t)}$-to-$\theta_n^{(t)}$ message in Fig. 2.1. Such a scheme was necessary both to prevent the propagation of an improper distribution, and also to prevent an exponential growth in the number of Gaussian terms that would be propagated using a Gaussian sum approximation. Due to the similarities between the MMV signal model of Section 2.2 and the DCS model of Section 3.2, such an approximation scheme is required once more.

To carry out the Gaussian sum approximation, we propose the following two schemes. The first is to simply choose a threshold τ that is slightly smaller than 1 and, using (C.35) as a guide, threshold $\overleftarrow{\pi}_n^{(t)}$ to choose between the two Gaussian components of (C.33). The resultant message is thus

$$\nu_{f_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)}) = \mathcal{CN}(\theta_n^{(t)}; \overrightarrow{\xi}_n^{(t)}, \overrightarrow{\psi}_n^{(t)}), \tag{3.6}$$

with $\overrightarrow{\xi}_n^{(t)}$ and $\overrightarrow{\psi}_n^{(t)}$ chosen according to

$$\left(\overrightarrow{\xi}_n^{(t)}, \overrightarrow{\psi}_n^{(t)}\right) = \begin{cases} \left(\frac{1}{\varepsilon}\phi_n^i, \frac{1}{\varepsilon^2}c_n^i\right), & \overleftarrow{\pi}_n^{(t)} \leq \tau \\ \left(\phi_n^i, c_n^i\right), & \overleftarrow{\pi}_n^{(t)} > \tau \end{cases}. \tag{3.7}$$

The second approach is to perform a second-order Taylor series approximation, as described in Section 2.4.2. The latter approach has the advantage of being parameter-free. Empirically, we find that this latter approach works well when changes in the

support occur infrequently, e.g., $p_{01} < 0.025$, while the former approach is better suited to more dynamic environments.

In Table 3.2 we provide a pseudo-code implementation of our proposed DCS-AMP algorithm that gives the explicit message update equations appropriate for performing a single forward pass. The interested reader can find an expanded derivation of the messages in Appendix C. The primary computational burden of DCS-AMP is computing the messages passing between the $\{x_n^{(t)}\}$ and $\{g_m^{(t)}\}$ nodes, a task which can be performed efficiently using matrix-vector products involving $\boldsymbol{A}^{(t)}$ and $\boldsymbol{A}^{(t)\mathsf{H}}$. The resulting overall complexity of DCS-AMP is therefore $\mathcal{O}(TMN)$ flops (flops-per-pass) when filtering (smoothing).[5] The storage requirements are $\mathcal{O}(N)$ and $\mathcal{O}(TN)$ complex numbers when filtering and smoothing, respectively.

3.4 Learning the signal model parameters

The signal model of Section 3.2 is specified by the Markov chain parameters λ, p_{01}, the Gauss-Markov parameters ζ, α, ρ, and the AWGN variance σ_e^2. It is likely that some or all of these parameters will require tuning in order to best match the unknown signal. As was the case in Section 2.5, we can use an EM algorithm to learn the relevant model parameters. The EM procedure is performed after each forward/backward pass, leading to a convergent sequence of parameter estimates. If operating in filtering mode, the procedure is similar, however the EM procedure is run after each recovered timestep using only causally available posterior estimates.

In Table 3.3, we provide the EM update equations for each of the parameters of our signal model, assuming DCS-AMP is operating in smoothing mode. Derivations for each update can be found in Appendix D.

[5] As with AMP-MMV, fast implicit operators capable of performing matrix-vector products will reduce DCS-AMP's complexity burden.

% Define soft-thresholding functions:

$$F_{nt}(\phi;c) \triangleq (1+\gamma_{nt}(\phi;c))^{-1}\Big(\frac{\overleftarrow{\psi}_n^{(t)}\phi+\overleftarrow{\xi}_n^{(t)}c}{\overleftarrow{\psi}_n^{(t)}+c}\Big) \tag{D5}$$

$$G_{nt}(\phi;c) \triangleq (1+\gamma_{nt}(\phi;c))^{-1}\Big(\frac{\overleftarrow{\psi}_n^{(t)}c}{\overleftarrow{\psi}_n^{(t)}+c}\Big) + \gamma_{nt}(\phi;c)|F_{nt}(\phi;c)|^2 \tag{D6}$$

$$F'_{nt}(\phi;c) \triangleq \tfrac{\partial}{\partial\phi}F_{nt}(\phi,c) = \tfrac{1}{c}G_{nt}(\phi;c) \tag{D7}$$

$$\gamma_{nt}(\phi;c) \triangleq \Big(\frac{1-\overleftarrow{\pi}_n^{(t)}}{\overleftarrow{\pi}_n^{(t)}}\Big)\Big(\frac{\overleftarrow{\psi}_n^{(t)}+c}{c}\Big) \times \exp\Big(-\Big[\frac{\overleftarrow{\psi}_n^{(t)}|\phi|^2+\overleftarrow{\xi}_n^{(t)*}c\phi+\overleftarrow{\xi}_n^{(t)}c\phi^*-c|\overleftarrow{\xi}_n^{(t)}|^2}{c(\overleftarrow{\psi}_n^{(t)}+c)}\Big]\Big) \tag{D8}$$

% Begin passing messages . . .
for $t = 1,\dots,T$:

% Execute the **(into)** phase . . .

$$\overleftarrow{\pi}_n^{(t)} = \frac{\overrightarrow{\lambda}_n^{(t)}\cdot\overleftarrow{\lambda}_n^{(t)}}{(1-\overrightarrow{\lambda}_n^{(t)})\cdot(1-\overleftarrow{\lambda}_n^{(t)})+\overrightarrow{\lambda}_n^{(t)}\cdot\overleftarrow{\lambda}_n^{(t)}} \quad \forall n \tag{A14}$$

$$\overleftarrow{\psi}_n^{(t)} = \frac{\overrightarrow{\kappa}_n^{(t)}\cdot\overleftarrow{\kappa}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}+\overleftarrow{\kappa}_n^{(t)}} \quad \forall n \tag{A15}$$

$$\overleftarrow{\xi}_n^{(t)} = \overleftarrow{\psi}_n^{(t)}\cdot\Big(\frac{\overrightarrow{\eta}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}}+\frac{\overleftarrow{\eta}_n^{(t)}}{\overleftarrow{\kappa}_n^{(t)}}\Big) \quad \forall n \tag{A16}$$

% Initialize AMP-related variables . . .

$\forall m: z_{mt}^1 = y_m^{(t)}, \forall n: \mu_{nt}^1 = 0,$ and $c_t^1 = 100\cdot\sum_{n=1}^N \psi_n^{(t)}$

% Execute the **(within)** phase using AMP . . .

for $i = 1,\dots,I,$:

$$\phi_{nt}^i = \textstyle\sum_{m=1}^M A_{mn}^{(t)*} z_{mt}^i + \mu_{nt}^i \quad \forall n \tag{A17}$$

$$\mu_{nt}^{i+1} = F_{nt}(\phi_{nt}^i; c_t^i) \quad \forall n \tag{A18}$$

$$v_{nt}^{i+1} = G_{nt}(\phi_{nt}^i; c_t^i) \quad \forall n \tag{A19}$$

$$c_t^{i+1} = \sigma_e^2 + \tfrac{1}{M}\textstyle\sum_{n=1}^N v_{nt}^{i+1} \tag{A20}$$

$$z_{mt}^{i+1} = y_m^{(t)} - \boldsymbol{a}_m^{(t)\mathsf{T}}\boldsymbol{\mu}_t^{i+1} + \tfrac{z_{mt}^i}{M}\textstyle\sum_{n=1}^N F'_{nt}(\phi_{nt}^i; c_t^i) \quad \forall m \tag{A21}$$

end

$$\hat{x}_n^{(t)} = \mu_{nt}^{I+1} \quad \forall n \qquad \text{% Store current estimate of } x_n^{(t)} \tag{A22}$$

% Execute the **(out)** phase . . .

$$\overrightarrow{\pi}_n^{(t)} = \Big(1+\Big(\frac{\overleftarrow{\pi}_n^{(t)}}{1-\overleftarrow{\pi}_n^{(t)}}\Big)\gamma_{nt}(\phi_{nt}^I; c_t^{I+1})\Big)^{-1} \quad \forall n \tag{A23}$$

$$(\overrightarrow{\xi}_n^{(t)}, \overrightarrow{\psi}_n^{(t)}) = \begin{cases} (\phi_n^I/\varepsilon, c_t^{I+1}/\varepsilon^2), & \overleftarrow{\pi}_n^{(t)} \le \tau \\ (\phi_n^I, c_t^{I+1}), & \text{o.w.} \end{cases} \quad \forall n \quad (\varepsilon \ll 1) \tag{A24}$$

% Execute the **(across)** phase forward in time . . .

$$\overrightarrow{\lambda}_n^{(t+1)} = \frac{p_{10}(1-\overrightarrow{\lambda}_n^{(t)})(1-\overrightarrow{\pi}_n^{(t)})+(1-p_{01})\overrightarrow{\lambda}_n^{(t)}\overrightarrow{\pi}_n^{(t)}}{(1-\overrightarrow{\lambda}_n^{(t)})(1-\overrightarrow{\pi}_n^{(t)})+\overrightarrow{\lambda}_n^{(t)}\overrightarrow{\pi}_n^{(t)}} \quad \forall n \tag{A25}$$

$$\overrightarrow{\eta}_n^{(t+1)} = (1-\alpha)\Big(\frac{\overrightarrow{\kappa}_n^{(t)}\overrightarrow{\psi}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}+\overrightarrow{\psi}_n^{(t)}}\Big)\Big(\frac{\overrightarrow{\eta}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}}+\frac{\overrightarrow{\xi}_n^{(t)}}{\overrightarrow{\psi}_n^{(t)}}\Big)+\alpha\zeta \quad \forall n \tag{A26}$$

$$\overrightarrow{\kappa}_n^{(t+1)} = (1-\alpha)^2\Big(\frac{\overrightarrow{\kappa}_n^{(t)}\overrightarrow{\psi}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}+\overrightarrow{\psi}_n^{(t)}}\Big)+\alpha^2\rho \quad \forall n \tag{A27}$$

end

Table 3.2: DCS-AMP steps for filtering mode, or the forward portion of a single forward/backward pass in smoothing mode. See Fig. 3.2 to associate quantities with the messages traversing the factor graph.

3.5 Incorporating Additional Structure

In Sections 3.2 - 3.4 we described a signal model for the dynamic CS problem and summarized a message passing algorithm for making inferences under this model,

% Define key quantities obtained from AMP-MMV at iteration k:

$$\mathrm{E}\big[s_n^{(t)}\big|\bar{\boldsymbol{y}}\big] = \frac{\big(\overrightarrow{\lambda}_n^{(t)}\overrightarrow{\pi}_n^{(t)}\overleftarrow{\lambda}_n^{(t)}\big)}{\big(\overrightarrow{\lambda}_n^{(t)}\overrightarrow{\pi}_n^{(t)}\overleftarrow{\lambda}_n^{(t)}+(1-\overrightarrow{\lambda}_n^{(t)})(1-\overrightarrow{\pi}_n^{(t)})(1-\overleftarrow{\lambda}_n^{(t)})\big)} \tag{Q1}$$

$$\mathrm{E}\big[s_n^{(t)}s_n^{(t-1)}\big|\bar{\boldsymbol{y}}\big] = p\big(s_n^{(t)}=1, s_n^{(t-1)}=1\big|\bar{\boldsymbol{y}}\big) \tag{Q2}$$

$$\tilde{v}_n^{(t)} \triangleq \mathrm{var}\{\theta_n^{(t)}|\bar{\boldsymbol{y}}\} = \left(\frac{1}{\overrightarrow{\kappa}_n^{(t)}}+\frac{1}{\overrightarrow{\psi}_n^{(t)}}+\frac{1}{\overleftarrow{\kappa}_n^{(t)}}\right)^{-1} \tag{Q3}$$

$$\tilde{\mu}_n^{(t)} \triangleq \mathrm{E}[\theta_n^{(t)}|\bar{\boldsymbol{y}}] = \tilde{v}_n^{(t)}\cdot\left(\frac{\overrightarrow{\eta}_n^{(t)}}{\overrightarrow{\kappa}_n^{(t)}}+\frac{\overrightarrow{\xi}_n^{(t)}}{\overrightarrow{\psi}_n^{(t)}}+\frac{\overleftarrow{\eta}_n^{(t)}}{\overleftarrow{\kappa}_n^{(t)}}\right) \tag{Q4}$$

$v_n^{(t)} \triangleq \mathrm{var}\big\{x_n^{(t)}\big|\bar{\boldsymbol{y}}\big\}$ % See (A19) of Table 3.2

$\mu_n^{(t)} \triangleq \mathrm{E}\big[x_n^{(t)}\big|\bar{\boldsymbol{y}}\big]$ % See (A18) of Table 3.2

% EM update equations:

$$\lambda^{k+1} = \tfrac{1}{N}\textstyle\sum_{n=1}^N \mathrm{E}\big[s_n^{(1)}\big|\bar{\boldsymbol{y}}\big] \tag{E6}$$

$$p_{01}^{k+1} = \frac{\sum_{t=2}^T\sum_{n=1}^N \mathrm{E}\big[s_n^{(t-1)}\big|\bar{\boldsymbol{y}}\big]-\mathrm{E}\big[s_n^{(t)}s_n^{(t-1)}\big|\bar{\boldsymbol{y}}\big]}{\sum_{t=2}^T\sum_{n=1}^N \mathrm{E}\big[s_n^{(t-1)}\big|\bar{\boldsymbol{y}}\big]} \tag{E7}$$

$$\begin{aligned}\zeta^{k+1} = &\left(\tfrac{N(T-1)}{\rho^k}+\tfrac{N}{(\sigma^2)^k}\right)^{-1}\Big(\tfrac{1}{(\sigma^2)^k}\textstyle\sum_{n=1}^N\tilde{\mu}_n^{(1)}\\ &+\textstyle\sum_{t=2}^T\sum_{n=1}^N\tfrac{1}{\alpha^k\rho^k}\big(\tilde{\mu}_n^{(t)}-(1-\alpha^k)\tilde{\mu}_n^{(t-1)}\big)\Big)\end{aligned} \tag{E8}$$

$$\alpha^{k+1} = \tfrac{1}{4N(T-1)}\Big(\mathfrak{b}-\sqrt{\mathfrak{b}^2+8N(T-1)\mathfrak{c}}\Big) \tag{E9}$$

where:

$$\begin{aligned}\mathfrak{b} \triangleq\ & \tfrac{2}{\rho^k}\textstyle\sum_{t=2}^T\sum_{n=1}^N\mathfrak{Re}\big\{E[\theta_n^{(t)*}\theta_n^{(t-1)}|\bar{\boldsymbol{y}}]\big\}\\ &-\mathfrak{Re}\{(\tilde{\mu}_n^{(t)}-\tilde{\mu}_n^{(t-1)})^*\zeta^k\}-\tilde{v}_n^{(t-1)}-|\tilde{\mu}_n^{(t-1)}|^2\\ \mathfrak{c} \triangleq\ & \tfrac{2}{\rho^k}\textstyle\sum_{t=2}^T\sum_{n=1}^N\tilde{v}_n^{(t)}+|\tilde{\mu}_n^{(t)}|^2+\tilde{v}_n^{(t-1)}+|\tilde{\mu}_n^{(t-1)}|^2\\ &-2\mathfrak{Re}\big\{E[\theta_n^{(t)*}\theta_n^{(t-1)}|\bar{\boldsymbol{y}}]\big\}\end{aligned}$$

$$\begin{aligned}\rho^{k+1} = &\tfrac{1}{(\alpha^k)^2N(T-1)}\textstyle\sum_{t=2}^T\sum_{n=1}^N\tilde{v}_n^{(t)}+|\tilde{\mu}_n^{(t)}|^2\\ &+(\alpha^k)^2|\zeta^k|^2-2(1-\alpha^k)\mathfrak{Re}\big\{E[\theta_n^{(t)*}\theta_n^{(t-1)}|\bar{\boldsymbol{y}}]\big\}\\ &-2\alpha^k\mathfrak{Re}\big\{\tilde{\mu}_n^{(t)*}\zeta^k\big\}+2\alpha^k(1-\alpha^k)\mathfrak{Re}\big\{\tilde{\mu}_n^{(t-1)*}\zeta^k\big\}\\ &+(1-\alpha^k)(\tilde{v}_n^{(t-1)}+|\tilde{\mu}_n^{(t-1)}|^2)\end{aligned} \tag{E10}$$

$$(\sigma_e^2)^{k+1} = \tfrac{1}{TM}\Big(\textstyle\sum_{t=1}^T\|\boldsymbol{y}^{(t)}-\boldsymbol{A}\boldsymbol{\mu}^{(t)}\|^2+\mathbf{1}_N^T\boldsymbol{v}^{(t)}\Big) \tag{E11}$$

Table 3.3: EM update equations for the signal model parameters of Section 3.2.

while iteratively learning the model parameters via EM. We also hinted that the model could be generalized to incorporate additional, or more complex, forms of structure. In this section we will elaborate on this idea, and illustrate one such generalization.

Recall that, in Section 3.2, we introduced hidden variables $\bar{\boldsymbol{s}}$ and $\bar{\boldsymbol{\theta}}$ in order to characterize the structure in the signal coefficients. An important consequence of introducing these hidden variables was that they made each signal coefficient $x_n^{(t)}$ conditionally independent of the remaining coefficients in $\bar{\boldsymbol{x}}$, given $s_n^{(t)}$ and $\theta_n^{(t)}$. This conditional independence served an important algorithmic purpose since it allowed us to apply the AMP algorithm, which requires independent local priors, within our larger inference procedure.

One way to incorporate additional structure into the signal model of Section 3.2 is to generalize our choices of $p(\bar{\boldsymbol{s}})$ and $p(\bar{\boldsymbol{\theta}})$. As a concrete example, pairing the temporal support model proposed in this chapter with the Markovian model of wavelet tree inter-scale correlations described in [16] through a more complex support prior, $p(\bar{\boldsymbol{s}})$, could enable even greater undersampling in a dynamic MRI setting. Performing inference on such models could be accomplished through the general algorithmic framework proposed in [61]. As another example, suppose that we wish to expand our Bernoulli-Gaussian signal model to one in which signal coefficients are marginally distributed according to a Bernoulli-Gaussian-mixture, i.e.,

$$p(x_n^{(t)}) = \lambda_0^{(t)}\delta(x_n^{(t)}) + \sum_{d=1}^{D} \lambda_d^{(t)} \mathcal{CN}(x_n^{(t)}; \zeta_d, \sigma_d^2),$$

where $\sum_{d=0}^{D} \lambda_d^{(t)} = 1$. Since we still wish to preserve the slow time-variations in the support and smooth evolution of non-zero amplitudes, a natural choice of hidden variables is $\{\bar{\boldsymbol{s}}, \bar{\boldsymbol{\theta}}_1, \ldots, \bar{\boldsymbol{\theta}}_D\}$, where $s_n^{(t)} \in \{0, 1, \ldots, D\}$, and $\theta_{d,n}^{(t)} \in \mathbb{C}$, $d = 1, \ldots, D$.

The relationship between $x_n^{(t)}$ and the hidden variables then generalizes to:

$$p(x_n^{(t)}|s_n^{(t)}, \theta_{1,n}^{(t)}, \ldots, \theta_{D,n}^{(t)}) = \begin{cases} \delta(x_n^{(t)}), & s_n^{(t)} = 0, \\ \delta(x_n^{(t)} - \theta_{d,n}^{(t)}), & s_n^{(t)} = d \neq 0. \end{cases}$$

To model the slowly changing support, we specify $p(\bar{\boldsymbol{s}})$ using a $(D+1)$-state Markov chain defined by the transition probabilities $p_{0d} \triangleq \Pr\{s_n^{(t)} = 0 | s_n^{(t-1)} = d\}$ and $p_{d0} \triangleq \Pr\{s_n^{(t)} = d | s_n^{(t-1)} = 0\}$, $d = 1, \ldots, D$. For simplicity, we assume that state transitions cannot occur between active mixture components, i.e., $\Pr(s_n^{(t)} = d | s_n^{(t-1)} = e) = 0$ when $d \neq e \neq 0$.[6] For each amplitude time-series we again use independent Gauss-Markov processes to model smooth evolutions in the amplitudes of active signal coefficients, i.e.,

$$\theta_{d,n}^{(t)} = (1 - \alpha_d)\big(\theta_{d,n}^{(t-1)} - \zeta_d\big) + \alpha_d w_{d,n}^{(t)} + \zeta_d,$$

where $w_{d,n}^{(t)} \sim \mathcal{CN}(0, \rho_d)$.

As a consequence of this generalized signal model, a number of the message computations of Section 3.3.2 must be modified. For steps **(into)** and **(across)**, it is largely straightforward to extend the computations to account for the additional hidden variables. For step **(within)**, the modifications will affect the AMP thresholding equations defined in (D5) - (D8) of Table 3.2. Details on a Bernoulli-Gaussian-mixture AMP algorithm can be found in [19]. For the **(out)** step, we will encounter difficulties applying standard sum-product update rules to compute the messages $\{\nu_{f_n^{(t)} \to \theta_{d,n}^{(t)}}(\cdot)\}_{d=1}^{D}$. As in the Bernoulli-Gaussian case, we consider a modification of

[6] By relaxing this restriction on active-to-active state transitions, we can model signals whose coefficients tend to enter the support set at small amplitudes that grow larger over time through the use of a Gaussian mixture component with a small variance that has a high probability of transitioning to a higher variance mixture component.

our assumed signal model that incorporates an $\varepsilon \ll 1$ term, and use Taylor series approximations of the resultant messages to collapse a $(D+1)$-ary Gaussian mixture to a single Gaussian. More information on this procedure can be found in Appendix B.

3.6 Numerical Study

We now describe the results of a numerical study of DCS-AMP.[7] The primary performance metric that we used in all of our experiments, which we refer to as the time-averaged normalized MSE (TNMSE), is defined as

$$\text{TNMSE}(\bar{\boldsymbol{x}}, \hat{\bar{\boldsymbol{x}}}) \triangleq \frac{1}{T}\sum_{t=1}^{T} \frac{\|\boldsymbol{x}^{(t)} - \hat{\boldsymbol{x}}^{(t)}\|_2^2}{\|\boldsymbol{x}^{(t)}\|_2^2},$$

where $\hat{\boldsymbol{x}}^{(t)}$ is an estimate of $\boldsymbol{x}^{(t)}$.

Unless otherwise noted, the following settings were used for DCS-AMP in our experiments. First, DCS-AMP was run as a smoother, with a total of 5 forward/backward passes. The number of inner AMP iterations I for each **(within)** step was $I = 25$, with a possibility for early termination if the change in the estimated signal, $\boldsymbol{\mu}_t^i$, fell below a predefined threshold from one iteration to the next, i.e., $\frac{1}{N}\|\boldsymbol{\mu}_t^i - \boldsymbol{\mu}_t^{i-1}\|_2 < 10^{-5}$. Equation (A22) of Table 3.2 was used to produce $\hat{\boldsymbol{x}}^{(t)}$, which corresponds to an MMSE estimate of $\boldsymbol{x}^{(t)}$ under DCS-AMP's estimated posteriors $p(x_n^{(t)}|\bar{\boldsymbol{y}})$. The amplitude approximation parameter ε from (C.33) was set to $\varepsilon = 10^{-7}$, while the threshold τ from (C.37) was set to $\tau = 0.99$. In our experiments, we found DCS-AMP's performance to be relatively insensitive to the value of ε provided that $\varepsilon \ll 1$. The choice of τ should be selected to provide a balance between allowing DCS-AMP to track amplitude evolutions on signals with rapidly varying supports and preventing DCS-AMP

[7] Code for reproducing our results is available at http://www.ece.osu.edu/~schniter/turboAMPdcs.

from prematurely gaining too much confidence in its estimate of the support. We found that the choice $\tau = 0.99$ works well over a broad range of problems. When the estimated transition probability $p_{01} < 0.025$, DCS-AMP automatically switched from the threshold method to the Taylor series method of computing (C.36), which is advantageous because it is parameter-free.

When learning model parameters adaptively from the data using the EM updates of Table 3.3, it is necessary to first initialize the parameters at reasonable values. Unless domain-specific knowledge suggests a particular initialization strategy, we advocate using the following simple heuristics: The initial sparsity rate, λ^1, active mean, ζ^1, active variance, $(\sigma^2)^1$, and noise variance, $(\sigma_e^2)^1$, can be initialized according to the procedure described in [19, §V].[8] The Gauss-Markov correlation parameter, α, can be initialized as

$$\alpha^1 = 1 - \frac{1}{T-1} \sum_{t=1}^{T-1} \frac{|\boldsymbol{y}^{(t)\,\mathsf{H}} \boldsymbol{y}^{(t+1)}|}{\lambda^1 (\sigma^2)^1 |\operatorname{tr}\{\boldsymbol{A}^{(t)} \boldsymbol{A}^{(t+1)\,\mathsf{H}}\}|}. \tag{3.8}$$

The active-to-inactive transition probability, p_{01}, is difficult to gauge solely from sample statistics involving the available measurements, $\bar{\boldsymbol{y}}$. We used $p_{01}^1 = 0.10$ as a generic default choice, based on the premise that it is easier for DCS-AMP to adjust to more dynamic signals once it has a decent "lock" on the static elements of the support, than it is for it to estimate relatively static signals under an assumption of high dynamicity.

[8] For problems with a high degree of undersampling and relatively non-sparse signals, it may be necessary to threshold the value for λ^1 suggested in [19] so that it does not fall below, e.g., 0.10.

3.6.1 Performance across the sparsity-undersampling plane

Two factors that have a significant effect on the performance of any CS algorithm are the sparsity $|\mathcal{S}^{(t)}|$ of the underlying signal, and the number of measurements M. Consequently, much can be learned about an algorithm by manipulating these factors and observing the resulting change in performance. To this end, we studied DCS-AMP's performance across the sparsity-undersampling plane [80], which is parameterized by two quantities, the *normalized sparsity ratio*, $\beta \triangleq \mathrm{E}[|\mathcal{S}^{(t)}|]/M$, and the *undersampling ratio*, $\delta \triangleq M/N$. For a given (δ, β) pair (with N fixed at 1500), sample realizations of $\bar{\boldsymbol{s}}$, $\bar{\boldsymbol{\theta}}$, and $\bar{\boldsymbol{e}}$ were drawn from their respective priors, and elements of a time-varying $\boldsymbol{A}^{(t)}$ were drawn from i.i.d. zero-mean complex circular Gaussians, with all columns subsequently scaled to have unit ℓ_2-norm, thus generating $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{y}}$.

As a performance benchmark, we used the support-aware Kalman smoother. In the case of linear dynamical systems with jointly Gaussian signal and observations, the Kalman filter (smoother) is known to provide MSE-optimal causal (non-causal) signal estimates [54]. When the signal is Bernoulli-Gaussian, the Kalman filter/smoother is no longer optimal. However, a lower bound on the achievable MSE can be obtained using the support-aware Kalman filter (SKF) or smoother (SKS). Since the classical state-space formulation of the Kalman filter does not easily yield the support-aware bound, we turn to an alternative view of Kalman filtering as an instance of message passing on an appropriate factor graph [81]. For this, it suffices to use the factor graph of Fig. 3.1 with $\{s_n^{(t)}\}$ treated as fixed, known quantities. Following the standard sum-product algorithm rules results in a message passing algorithm in which all messages are Gaussian, and no message approximations are required. Then, by running loopy Gaussian belief propagation until convergence, we are guaranteed that the resultant posterior means constitute the MMSE estimate of $\bar{\boldsymbol{x}}$ [68, Claim 5].

To quantify the improvement obtained by exploiting temporal correlation, signal

recovery was also explored using the Bernoulli-Gaussian AMP algorithm (BG-AMP) independently at each timestep (i.e., ignoring temporal structure in the support and amplitudes), accomplished by passing messages only within the dashed boxes of Fig. 3.1 using $p\big(x_n^{(t)}\big)$ from (3.4) as AMP's prior.[9]

In Fig. 3.3, we present four plots from a representative experiment. The TNMSE across the (logarithmically scaled) sparsity-undersampling plane is shown for (working from left to right) the SKS, DCS-AMP, EM-DCS-AMP (DCS-AMP with EM parameter tuning), and BG-AMP. In order to allow EM-DCS-AMP ample opportunity to converge to the correct parameter values, it was allowed up to 300 EM iterations/smoothing passes, although it would quite often terminate much sooner if the parameter initializations were reasonably close. The results shown were averaged over more than 300 independent trials at each (δ, β) pair. For this experiment, signal model parameters were set at $N = 1500$, $T = 25$, $p_{01} = 0.05$, $\zeta = 0$, $\alpha = 0.01$, $\sigma^2 = 1$, and a noise variance, σ_e^2, chosen to yield a signal-to-noise ratio (SNR) of 25 dB. (M, λ) were set based on specific (δ, β) pairs, and p_{10} was set so as to keep the expected number of active coefficients constant across time. It is interesting to observe that the performance of the SKS and (EM-)DCS-AMP are only weakly dependent on the undersampling ratio δ. In contrast, the structure-agnostic BG-AMP algorithm is strongly affected. This is one of the principal benefits of incorporating temporal structure; it makes it possible to tolerate more substantial amounts of undersampling, particularly when the underlying signal is less sparse.

[9] Experiments were also run that compared performance against Basis Pursuit Denoising (BPDN) [82] with genie-aided parameter tuning (solved using the SPGL1 solver [83]). However, this was found to yield higher TNMSE than BG-AMP, and at higher computational cost.

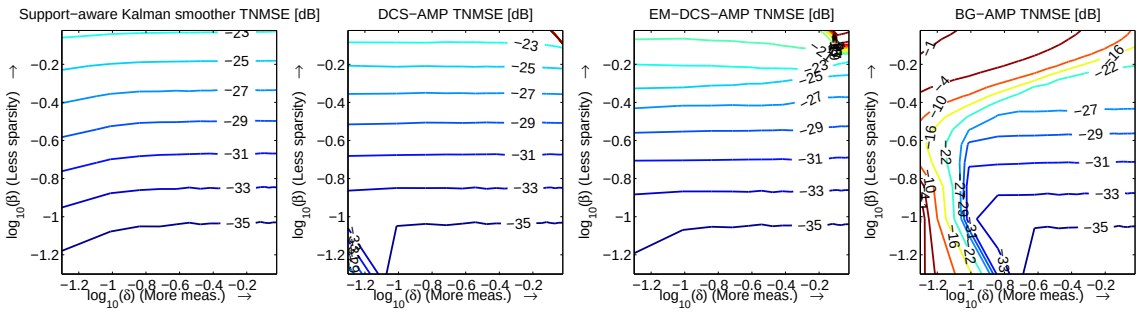


Figure 3.3: A plot of the TNMSE (in dB) of (from left) the SKS, DCS-AMP, EM-DCS-AMP, and BG-AMP across the sparsity-undersampling plane, for temporal correlation parameters $p_{01} = 0.05$ and $\alpha = 0.01$.

3.6.2 Performance vs p_{01} and α

The temporal correlation of our time-varying sparse signal model is largely dictated by two parameters, the support transition probability p_{01} and the amplitude forgetting factor α. Therefore, it is worth investigating how the performance of (EM-)DCS-AMP is affected by these two parameters. In an experiment similar to that of Fig. 3.3, we tracked the performance of (EM-)DCS-AMP, the SKS, and BG-AMP across a plane of (p_{01}, α) pairs. The active-to-inactive transition probability p_{01} was swept linearly over the range $[0, 0.15]$, while the Gauss-Markov amplitude forgetting factor α was swept logarithmically over the range $[0.001, 0.95]$. To help interpret the meaning of these parameters, we note that the fraction of the support that is expected to change from one timestep to the next is given by $2\,p_{01}$, and that the Pearson correlation coefficient between temporally adjacent amplitude variables is $1 - \alpha$.

In Fig. 3.4 we plot the TNMSE (in dB) of the SKS and (EM-)DCS-AMP as a function of the percentage of the support that changes from one timestep to the next (i.e., $2p_{01} \times 100$) and the logarithmic value of α for a signal model in which $\delta = 1/5$ and $\beta = 0.60$, with remaining parameters set as before. Since BG-AMP is agnostic

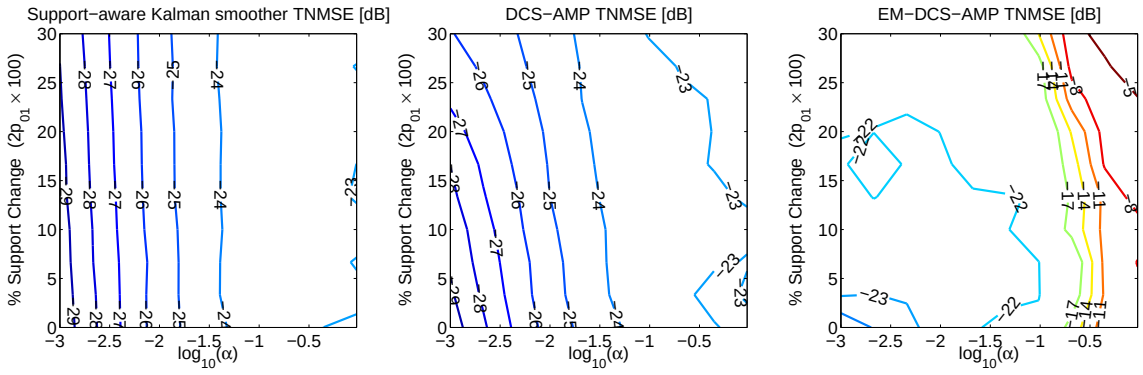


Figure 3.4: TNMSE (in dB) of (from left) the SKS, DCS-AMP, and EM-DCS-AMP as a function of the model parameters p_{01} and α, for undersampling ratio $\delta = 1/3$ and sparsity ratio $\beta = 0.45$. BG-AMP achieved a TNMSE of -5.9 dB across the plane.

to temporal correlation, its performance is insensitive to the values of p_{01} and α. Therefore, we do not include a plot of the performance of BG-AMP, but note that it achieved a TNMSE of -5.9 dB across the plane. For the SKS and (EM-)DCS-AMP, we see that performance improves with increasing amplitude correlation (moving leftward). This behavior, although intuitive, is in contrast to the relationship between performance and correlation found in the MMV problem [14, 32], primarily due to the fact that the measurement matrix is static for all timesteps in the classical MMV problem, whereas here it varies with time. Since the SKS has perfect knowledge of the support, its performance is only weakly dependent on the rate of support change. DCS-AMP performance shows a modest dependence on the rate of support change, but nevertheless is capable of managing rapid temporal changes in support, while EM-DCS-AMP performs very near the level of the noise when $\alpha < 0.10$.

3.6.3 Recovery of an MRI image sequence

While the above simulations demonstrate the effectiveness of DCS-AMP in recovering signals generated according to our signal model, it remains to be seen whether the signal model itself is suitable for describing practical dynamic CS signals. To address this question, we tested the performance of DCS-AMP on a dynamic MRI experiment first performed in [8]. The experiment consists of recovering a sequence of 10 MRI images of the larynx, each 256×256 pixels in dimension. (See Fig. 3.5.) The measurement matrices were never stored explicitly due to the prohibitive sizes involves, but were instead treated as the composition of three linear operations, $\boldsymbol{A} = \boldsymbol{M}\boldsymbol{F}\boldsymbol{W}^{\mathsf{T}}$. The first operation, $\boldsymbol{W}^{\mathsf{T}}$, was the synthesis of the underlying image from a sparsifying 2-D, 2-level Daubechies-4 wavelet transform representation. The second operation, $\boldsymbol{F}$, was a 2-D Fourier transform that yielded the k-space coefficients of the image. The third operation, $\boldsymbol{M}$, was a sub-sampling mask that kept only a fraction of the available k-space data.

Since the image transform coefficients are compressible rather than sparse, the SKF/SKS no longer serves as an appropriate algorithmic benchmark. Instead, we compare performance against Modified-CS [48], as well as timestep-independent Basis Pursuit.[10] As reported in [48], Modified-CS demonstrates that substantial improvements can be obtained over temporally agnostic methods.

Since the statistics of wavelet coefficients at different scales are often highly dissimilar (e.g., the coarsest-scale approximation coefficients are usually much less sparse than those at finer scales, and are also substantially larger in magnitude), we allowed our EM procedure to learn different parameters for different wavelet scales. Using

[10] Modified-CS is available at `http://home.engineering.iastate.edu/~luwei/modcs/index.html`. Basis Pursuit was solved using the ℓ_1-MAGIC equality-constrained primal-dual solver (chosen since it is used as a subroutine within Modified-CS), available at `http://users.ece.gatech.edu/~justin/l1magic/`.

$\mathcal{G}_1$ to denote the indices of the coarsest-scale "approximation" coefficients, and $\mathcal{G}_2$ to denote the finer-scale "wavelet" coefficients, DCS-AMP was initialized with the following parameter choices: $\lambda_{\mathcal{G}_1} = 0.99$, $\lambda_{\mathcal{G}_2} = 0.01$, $p_{01} = 0.01$, $\zeta_{\mathcal{G}_1} = \zeta_{\mathcal{G}_2} = 0$, $\alpha_{\mathcal{G}_1} = \alpha_{\mathcal{G}_2} = 0.05$, $\rho_{\mathcal{G}_1} = 10^5$, $\rho_{\mathcal{G}_2} = 10^4$, and $\sigma_e^2 = 0.01$, and run in filtering mode with $I = 10$ inner AMP iterations.

We note that our initializations were deliberately chosen to be agnostic, but reasonable, values. In particular, observing that the coarsest-scale approximation coefficients of a wavelet decomposition are almost surely non-zero, we initialized the associated group's sparsity rate at $\lambda_{\mathcal{G}_1} = 0.99$, while the finer scale detail coefficients were given an arbitrary sparsity-promoting rate of $\lambda_{\mathcal{G}_2} = 0.01$. The choices of α and ρ were driven by an observation that the variance of coefficients across wavelet scales often differs by an order-of-magnitude. The noise variance is arguably the most important parameter to initialize properly, since it balances the conflicting objectives of fitting the data and adhering to the assumed signal model. Our rule-of-thumb for initializing this parameter was that it is best to err on the side of fitting the data (since the SNR in this MRI data collection was high), and thus we initialized the noise variance with a small value.

In Table 3.4 we summarize the performance of three different estimators: timestep-independent Basis Pursuit, which performs independent ℓ_1 minimizations at each timestep, Modified-CS, and DCS-AMP (operating in filtering mode). In this experiment, per the setup described in [8], the initial timestep was sampled at 50% of the Nyquist rate, i.e., $M = N/2$, while subsequent timesteps were sampled at 16% of the Nyquist rate. Both Modified-CS and DCS-AMP substantially outperform Basis Pursuit with respect to TNMSE, with DCS-AMP showing a slight advantage over Modified-CS. Despite the similar TNMSE performance, note that DCS-AMP runs in seconds, whereas Modified-CS takes multiple hours. In Fig. 3.5, we plot the true

Algorithm	TNMSE (dB)	Runtime
Basis Pursuit	-17.22	47 min
Modified-CS	-34.30	7.39 hrs
DCS-AMP (Filter)	**-34.62**	**8.08 sec**

Table 3.4: Performance on dynamic MRI dataset from [8] with increased sampling rate at initial timestep.

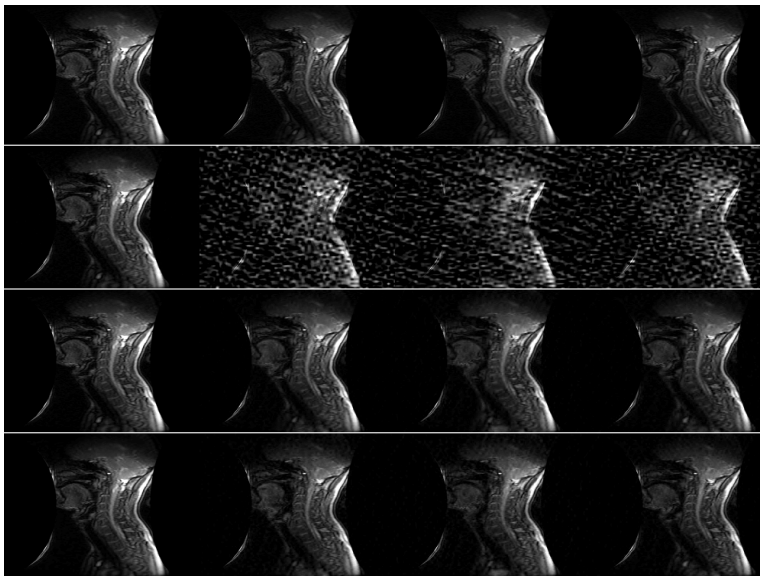


Figure 3.5: Frames 1, 2, 5, and 10 of the dynamic MRI image sequence of (from top to bottom): the fully sampled dataset, Basis Pursuit, Modified-CS, and DCS-AMP, with increased sampling rate at initial timestep.

images along with the recoveries for this experiment, which show severe degradation for Basis Pursuit on all but the initial timestep.

In practice, it may not be possible to acquire an increased number of samples at the initial timestep. We therefore repeated the experiment while sampling at 16% of the Nyquist rate at every timestep. The results, shown in Table 3.5, show that DCS-AMP's performance degrades by about 5 dB, while Modified-CS suffers a

Algorithm	TNMSE (dB)	Runtime
Basis Pursuit	-16.83	47.61 min
Modified-CS	-17.18	7.78 hrs
DCS-AMP (Filter)	**-29.51**	**7.27 sec**

Table 3.5: Performance on dynamic MRI dataset from [8] with identical sampling rate at every timestep.

14 dB reduction, illustrating that, when the estimate of the initial support is poor, Modified-CS struggles to outperform Basis Pursuit.

3.6.4 Recovery of a CS audio sequence

In another experiment using real-world data, we used DCS-AMP to recover an audio signal from sub-Nyquist samples. In this case, we employ the Bernoulli-Gaussian-mixture signal model proposed for DCS-AMP in Section 3.5. The audio clip is a 7 second recording of a trumpet solo, and contains a succession of rapid changes in the trumpet's pitch. Such a recording presents a challenge for CS methods, since the signal will be only compressible, and not sparse. The clip, sampled at a rate of 11 kHz, was divided into $T = 54$ non-overlapping segments of length $N = 1500$. Using the discrete cosine transform (DCT) as a sparsifying basis, linear measurements were obtained using a time-invariant i.i.d. Gaussian sensing matrix.

In Fig. 3.6 we plot the magnitude of the DCT coefficients of the audio signal on a dB scale. Beyond the temporal correlation evident in the plot, it is also interesting to observe that there is a non-trivial amount of frequency correlation (correlation across the index $[n]$), as well as a large dynamic range. We performed recoveries using four techniques: BG-AMP, GM-AMP (a temporally agnostic Bernoulli-Gaussian-mixture AMP algorithm with $D = 4$ Gaussian mixture components), DCS-(BG)-AMP, and

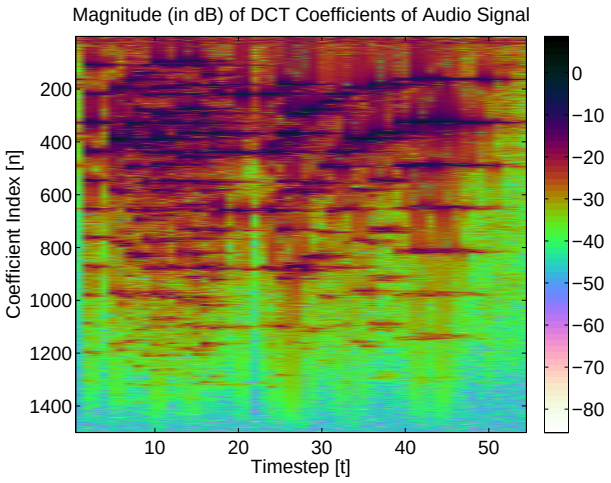


Figure 3.6: DCT coefficient magnitudes (in dB) of an audio signal.

DCS-GM-AMP (the Bernoulli-Gaussian-mixture dynamic CS model described in Section 3.5, with $D = 4$). For each algorithm, EM learning of the model parameters was performed using straightforward variations of the procedure described in Section 3.4, with model parameters initialized automatically using simple heuristics described in [19]. Moreover, unique model parameters were learned at each timestep (with the exception of support transition probabilities). Furthermore, since our model of hidden amplitude evolutions was poorly matched to this audio signal, we fixed $\alpha = 1$.

In Table 3.6 we present the results of applying each algorithm to the audio dataset for three different undersampling rates, δ. For each algorithm, both the TNMSE in dB and the runtime in seconds are provided. Overall, we see that performance improves at each undersampling rate as the signal model becomes more expressive. While GM-AMP outperforms BG-AMP at all undersampling rates, it is surpassed by DCS-BG-AMP and DCS-GM-AMP, with DCS-GM-AMP offering the best TNMSE performance. Indeed, we observe that one can obtain comparable, or even better,

		Undersampling Rate		
		$\delta = \frac{1}{2}$	$\delta = \frac{1}{3}$	$\delta = \frac{1}{5}$
Algorithm	BG-AMP	-16.88 (dB) **09.11 (s)**	-11.67 (dB) **08.27 (s)**	-08.56 (dB) **06.63 (s)**
	GM-AMP ($D = 4$)	-17.49 (dB) 19.36 (s)	-13.74 (dB) 17.48 (s)	-10.23 (dB) 15.98 (s)
	DCS-BG-AMP	-19.84 (dB) 10.20 (s)	-14.33 (dB) 08.39 (s)	-11.40 (dB) 06.71 (s)
	DCS-GM-AMP ($D = 4$)	**-21.33 (dB)** 20.34 (s)	**-16.78 (dB)** 18.63 (s)	**-12.49 (dB)** 10.13 (s)

Table 3.6: Performance on audio CS dataset (TNMSE (dB) | Runtime (s)) of two temporally independent algorithms, BG-AMP and GM-AMP, and two temporally structured algorithms, DCS-BG-AMP and DCS-GM-AMP.

performance with an undersampling rate $\delta = \frac{1}{5}$ using DCS-BG-AMP or DCS-GM-AMP, with that obtained using BG-AMP with an undersampling rate $\delta = \frac{1}{3}$.

3.6.5 Frequency Estimation

In a final experiment, we compared the performance of DCS-AMP against techniques designed to solve the problem of subspace identification and tracking from partial observations (SITPO) [84, 85], which bears similarities to the dynamic CS problem. In subspace identification, the goal is to learn the low-dimensional subspace occupied by multi-timestep data measured in a high ambient dimension, while in subspace tracking, the goal is to track that subspace as it evolves over time. In the partial observation setting, the high-dimensional observations are sub-sampled using a mask that varies with time. The dynamic CS problem can be viewed as a special case of SITPO, wherein the time-t subspace is spanned by a subset of the columns of an a priori known matrix $\boldsymbol{A}^{(t)}$. One problem that lies in the intersection of SITPO and dynamic CS is frequency tracking from partial time-domain observations.

For comparison purposes, we replicated the "direction of arrival analysis" experiment described in [85] where the observations at time t take the form

$$\boldsymbol{y}^{(t)} = \boldsymbol{\Phi}^{(t)}\boldsymbol{V}^{(t)}\boldsymbol{a}^{(t)} + \boldsymbol{e}^{(t)}, \quad t = 1, 2, \ldots, T \tag{3.9}$$

where $\boldsymbol{\Phi}^{(t)} \in \{0,1\}^{M\times N}$ is a selection matrix with non-zero column indices $\mathcal{Q}^{(t)} \subset \{1,\ldots,N\}$, $\boldsymbol{V}^{(t)} \in \mathbb{C}^{N\times K}$ is a Vandermonde matrix of sampled complex sinusoids, i.e.,

$$\boldsymbol{V}^{(t)} \triangleq [\boldsymbol{v}(\omega_1^{(t)}), \ldots, \boldsymbol{v}(\omega_K^{(t)})], \tag{3.10}$$

with $\boldsymbol{v}(\omega_k^{(t)}) \triangleq [1, e^{j2\pi\omega_k^{(t)}}, \ldots, e^{j2\pi\omega_k^{(t)}(N-1)}]^{\mathsf{T}}$ and $\omega_k^{(t)} \in [0,1)$. $\boldsymbol{a}^{(t)} \in \mathbb{R}^K$ is a vector of instantaneous amplitudes, and $\boldsymbol{e}^{(t)} \in \mathbb{R}^N$ is additive noise with i.i.d. $\mathcal{N}(0,\sigma_e^2)$ elements.[11] Here, $\{\boldsymbol{\Phi}^{(t)}\}_{t=1}^T$ is known, while $\{\boldsymbol{\omega}^{(t)}\}_{t=1}^T$ and $\{\boldsymbol{a}^{(t)}\}_{t=1}^T$ are unknown, and our goal is to estimate them. To assess performance, we report TNMSE in the estimation of the "complete" signal $\{\boldsymbol{V}^{(t)}\boldsymbol{a}^{(t)}\}_{t=1}^T$.

We compared DCS-AMP's performance against two online algorithms designed to solve the SITPO problem: GROUSE [84] and PETRELS [85]. Both GROUSE and PETRELS return time-varying subspace estimates, which were passed to an ESPRIT algorithm to generate time-varying frequency estimates (as in [85]). Finally, time-varying amplitude estimates were computed using least-squares. For DCS-AMP, we constructed $\boldsymbol{A}^{(t)}$ using a 2× column-oversampled DFT matrix, keeping only those rows indexed by $\mathcal{Q}^{(t)}$. DCS-AMP was run in filtering mode for fair comparison with the "online" operation of GROUSE and PETRELS, with $I = 7$ inner AMP iterations.

[11] Code for replicating the experiment provided by the authors of [85]. Unless otherwise noted, specific choices regarding $\{\omega_k^{(t)}\}$ and $\{\boldsymbol{a}^{(t)}\}$ were made by the authors of [85] in a deterministic fashion, and can be found in the code.

The results of performing the experiment for three different problem configurations are presented in Table 3.7, with performance averaged over 100 independent realizations. All three algorithms were given the true value of K. In the first problem setup considered, we see that GROUSE operates the fastest, although its TNMSE performance is noticeably inferior to that of both PETRELS and DCS-AMP, which provide similar TNMSE performance and complexity. In the second problem setup, we reduce the number of measurements, M, from 30 to 10, leaving all other settings fixed. In this regime, both GROUSE and PETRELS are unable to accurately estimate $\{\omega_k^{(t)}\}$, and consequently fail to accurately recover $\boldsymbol{V}^{(t)}\boldsymbol{a}^{(t)}$, in contrast to DCS-AMP. In the third problem setup, we increased the problem dimensions from the first problem setup by a factor of 4 to understand how the complexity of each approach scales with problem size. In order to increase the number of "active" frequencies from $K = 5$ to $K = 20$, 15 additional frequencies and amplitudes were added uniformly at random to the 5 deterministic trajectories of the preceding experiments. Interestingly, DCS-AMP, which was the slowest at smaller problem dimensions, becomes the fastest (and most accurate) in the higher-dimensional setting, scaling much better than either GROUSE or PETRELS.

3.7 Conclusion

In this chapter we proposed DCS-AMP, a novel approach to dynamic CS. Our technique merges ideas from the fields of belief propagation and switched linear dynamical systems, together with a computationally efficient inference method known as AMP. Moreover, we proposed an EM approach that learns all model parameters automatically from the data. In numerical experiments on synthetic data, DCS-AMP performed within 3 dB of the support-aware Kalman smoother bound across the sparsity-undersampling plane. Repeating the dynamic MRI experiment from [8],

		Problem Setup		
		$(N, M, K) = (256, 30, 5)$	$(N, M, K) = (256, 10, 5)$	$(N, M, K) = (1024, 120, 20)$
Algorithm	GROUSE	-4.52 (dB) **6.78 (s)**	2.02 (dB) **6.68 (s)**	-4.51 (dB) 173.89 (s)
	PETRELS	**-15.62** (dB) 29.51 (s)	0.50 (dB) 14.93 (s)	-7.98 (dB) 381.10 (s)
	DCS-AMP	-15.46 (dB) 34.49 (s)	**-10.85 (dB)** 28.42 (s)	**-12.79 (dB)** **138.07 (s)**

Table 3.7: Average performance on synthetic frequency estimation experiment (TNMSE (dB) | Runtime (s)) of GROUSE, PETRELS, and DCS-AMP. In all cases, $T = 4000$, $\sigma_e^2 = 10^{-6}$.

DCS-AMP slightly outperformed Modified-CS in MSE, but required less than 10 seconds to run, in comparison to more than 7 hours for Modified-CS. For the compressive sensing of audio, we demonstrated significant gains from the exploitation of temporal structure and Gaussian-mixture learning of the signal prior. Lastly, we found that DCS-AMP can outperform recent approaches to Subspace Identification and Tracking from Partial Observations (SITPO) when the underlying problem can be well-represented through a dynamic CS model.

CHAPTER 4

BINARY CLASSIFICATION, FEATURE SELECTION, AND MESSAGE PASSING

"It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."

- Sherlock Holmes

Chapters 2 and 3 examined particular instances of sparse linear regression problems. A complementary problem to that of sparse linear regression is the problem of binary linear classification and feature selection [67], which is the subject of this chapter.[1,2]

4.1 Introduction

The objective of *binary linear classification* is to learn the weight vector $\mathbf{w} \in \mathbb{R}^N$ that best predicts an unknown binary class label $\mathsf{y} \in \{-1, 1\}$ associated with a given

[1] Work presented in this chapter is largely excerpted from a manuscript co-authored with Per Sederberg and Philip Schniter, entitled "Binary Linear Classification and Feature Selection via Generalized Approximate Message Passing." [86]

[2] A caution to the reader: To conform to the convention adopted in classification literature, in this chapter we use $\boldsymbol{w}$ to denote the unknown vector we wish to infer (instead of $\boldsymbol{x}$), and the matrix $\boldsymbol{X}$ assumes the role of $\boldsymbol{A}$ in Chapters 2 and 3.

vector of quantifiable features $\boldsymbol{x} \in \mathbb{R}^N$ from the sign of a linear "score" $\mathsf{z} \triangleq \langle \boldsymbol{x}, \mathbf{w} \rangle$.[3] The goal of *linear feature selection* is to identify which subset of the N weights in $\mathbf{w}$ are necessary for accurate prediction of the unknown class label y, since in some applications (e.g., multi-voxel pattern analysis) this subset itself is of primary concern.

In formulating this linear feature selection problem, we assume that there exists a K-*sparse* weight vector $\mathbf{w}$ (i.e., $\|\mathbf{w}\|_0 = K \ll N$) such that $\mathsf{y} = \operatorname{sgn}(\langle \boldsymbol{x}, \mathbf{w} \rangle - \mathsf{e})$, where $\operatorname{sgn}(\cdot)$ is the signum function and $\mathsf{e} \sim p_{\mathsf{e}}$ is a random perturbation accounting for model inaccuracies. For the purpose of learning $\mathbf{w}$, we assume the availability of M labeled training examples generated independently according to this model:

$$y_m = \operatorname{sgn}(\langle \boldsymbol{x}_m, \mathbf{w} \rangle - \mathsf{e}_m), \quad \forall\, m = 1, \ldots, M, \tag{4.1}$$

with $\mathsf{e}_m \sim$ i.i.d p_{e}. It is common to express the relationship between the label y_m and the score $\mathsf{z}_m \triangleq \langle \boldsymbol{x}_m, \mathbf{w} \rangle$ in (4.1) via the conditional pdf $p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|z_m)$, known as the "activation function," which can be related to the perturbation pdf p_{e} via

$$p_{\mathsf{y}_m|\mathsf{z}_m}(1|z_m) = \int_{-\infty}^{z_m} p_{\mathsf{e}}(e)\, de = 1 - p_{\mathsf{y}_m|\mathsf{z}_m}(-1|z_m). \tag{4.2}$$

We are particularly interested in classification problems in which the number of potentially discriminatory features N drastically exceeds the number of available training examples M. Such computationally challenging problems are of great interest in a number of modern applications, including text classification [87], multi-voxel pattern analysis (MVPA) [88–90], conjoint analysis [91], and microarray gene expression [92]. In MVPA, for instance, neuroscientists attempt to infer which regions

[3] We note that one could also compute the score from a fixed non-linear transformation $\psi(\cdot)$ of the original feature $\boldsymbol{x}$ via $\mathsf{z} \triangleq \langle \psi(\boldsymbol{x}), \mathbf{w} \rangle$ as in kernel-based classification. Although the methods we describe here are directly compatible with this approach, we write $\mathsf{z} = \langle \boldsymbol{x}, \mathbf{w} \rangle$ for simplicity.

in the human brain are responsible for distinguishing between two cognitive states by measuring neural activity via fMRI at $N \approx 10^4$ voxels. Due to the expensive and time-consuming nature of working with human subjects, classifiers are routinely trained using only $M \approx 10^2$ training examples, and thus $N \gg M$.

In the $N \gg M$ regime, the model of (4.1) coincides with that of *noisy one-bit compressed sensing* (CS) [93, 94]. In that setting, it is typical to write (4.1) in matrix-vector form using $\boldsymbol{y} \triangleq [y_1, \ldots, y_M]^{\mathsf{T}}$, $\mathbf{e} \triangleq [\mathsf{e}_1, \ldots, \mathsf{e}_M]^{\mathsf{T}}$, $\boldsymbol{X} \triangleq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M]^{\mathsf{T}}$, and elementwise $\mathrm{sgn}(\cdot)$, yielding

$$\boldsymbol{y} = \mathrm{sgn}(\boldsymbol{X}\mathbf{w} - \mathbf{e}), \tag{4.3}$$

where $\mathbf{w}$ embodies the signal-of-interest's sparse representation, $\boldsymbol{X} = \boldsymbol{\Phi}\boldsymbol{\Psi}$ is a concatenation of a linear measurement operator $\boldsymbol{\Phi}$ and a sparsifying signal dictionary $\boldsymbol{\Psi}$, and $\mathbf{e}$ is additive noise.[4] Importantly, in the $N \gg M$ setting, [94] established performance guarantees on the estimation of K-sparse $\mathbf{w}$ from $O(K \log N/K)$ binary measurements of the form (4.3), under i.i.d Gaussian $\{\boldsymbol{x}_m\}$ and mild conditions on the perturbation process $\{\mathsf{e}_m\}$, even when the entries within $\boldsymbol{x}_m$ are correlated. This result implies that, in large binary linear classification problems, accurate feature selection is indeed possible from $M \ll N$ training examples, as long as the underlying weight vector $\mathbf{w}$ is sufficiently sparse. Not surprisingly, many techniques have been proposed to find such weight vectors [24, 95–101].

In addition to theoretical analyses, the CS literature also offers a number of high-performance algorithms for the inference of $\mathbf{w}$ in (4.3), e.g., [93,94,102–105]. Thus, the question arises as to whether these algorithms also show advantages in the domain

[4]For example, the common case of additive white Gaussian noise (AWGN) $\{\mathsf{e}_m\} \sim$ i.i.d $\mathcal{N}(0, v)$ corresponds to the "probit" activation function, i.e., $p_{\mathsf{y}_m|\mathsf{z}_m}(1|z_m) = \Phi(z_m/v)$, where $\Phi(\cdot)$ is the standard-normal cdf.

of binary linear classification and feature selection. In this paper, we answer this question in the affirmative by focusing on the *generalized approximate message passing* (GAMP) algorithm [27], which extends the AMP algorithm [25, 26] from the case of linear, AWGN-corrupted observations (i.e., $\boldsymbol{y} = \boldsymbol{X}\mathbf{w} - \mathbf{e}$ for $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, v\boldsymbol{I})$) to the case of generalized-linear observations, such as (4.3). AMP and GAMP are attractive for several reasons: *(i)* For i.i.d sub-Gaussian $\boldsymbol{X}$ in the large-system limit (i.e., $M, N \to \infty$ with fixed ratio $\delta = \frac{M}{N}$), they are rigorously characterized by a state-evolution whose fixed points, when unique, are optimal [30]; *(ii)* Their state-evolutions predict fast convergence rates and per-iteration complexity of only $O(MN)$; *(iii)* They are very flexible with regard to data-modeling assumptions (see, e.g., [61]); *(iv)* Their model parameters can be learned online using an expectation-maximization (EM) approach that has been shown to yield state-of-the-art mean-squared reconstruction error in CS problems [106].

In this chapter, we develop a GAMP-based approach to binary linear classification and feature selection that makes the following contributions: 1) in Section 4.2, we show that GAMP implements a particular approximation to the error-rate minimizing linear classifier under the assumed model (4.1); 2) in Section 4.3, we show that GAMP's state evolution framework can be used to characterize the misclassification rate in the large-system limit; 3) in Section 4.4, we develop methods to implement logistic, probit, and hinge-loss-based regression using both max-sum and sum-product versions of GAMP, and we further develop a method to make these classifiers robust in the face of corrupted training labels; and 4) in Section 4.5, we present an EM-based scheme to learn the model parameters online, as an alternative to cross-validation. The numerical study presented in Section 4.6 then confirms the efficacy, flexibility, and speed afforded by our GAMP-based approaches to binary classification and feature selection.

4.1.1 Notation

Random quantities are typeset in sans-serif (e.g., e) while deterministic quantities are typeset in serif (e.g., e). The pdf of random variable e under deterministic parameters $\boldsymbol{\theta}$ is written as $p_{\mathsf{e}}(e;\boldsymbol{\theta})$, where the subscript and parameterization are sometimes omitted for brevity. Column vectors are typeset in boldface lower-case (e.g., $\boldsymbol{y}$ or $\mathbf{y}$), matrices in boldface upper-case (e.g., $\boldsymbol{X}$ or $\mathbf{X}$), and their transpose is denoted by $(\cdot)^{\mathsf{T}}$. For vector $\boldsymbol{y} = [y_1, \ldots, y_N]^{\mathsf{T}}$, $\boldsymbol{y}_{m:n}$ refers to the subvector $[y_m, \ldots, y_n]^{\mathsf{T}}$. Finally, $\mathcal{N}(\boldsymbol{a}; \boldsymbol{b}, \boldsymbol{C})$ is the multivariate normal distribution as a function of $\boldsymbol{a}$, with mean $\boldsymbol{b}$, and with covariance matrix $\boldsymbol{C}$, while $\phi(\cdot)$ and $\Phi(\cdot)$ denote the standard normal pdf and cdf, respectively.

4.2 Generalized Approximate Message Passing

In this section, we introduce generalized approximate message passing (GAMP) from the perspective of binary linear classification. In particular, we show that the *sum-product* variant of GAMP is a loopy belief propagation (LBP) approximation of the classification-error-rate minimizing linear classifier and that the *max-sum* variant of GAMP is a LBP implementation of the standard regularized-loss-minimization approach to linear classifier design.

4.2.1 Sum-Product GAMP

Suppose that we are given M labeled training examples $\{y_m, \boldsymbol{x}_m\}_{m=1}^{M}$, and T test feature vectors $\{\boldsymbol{x}_t\}_{t=M+1}^{M+T}$ associated with unknown test labels $\{\mathsf{y}_t\}_{t=M+1}^{M+T}$, all obeying the noisy linear model (4.1) under some known error pdf p_{e}, and thus known $p_{\mathsf{y}_m|\mathsf{z}_m}$.

We then consider the problem of computing the classification-error-rate minimizing hypotheses $\{\hat{y}_t\}_{t=M+1}^{M+T}$,

$$\hat{y}_t = \underset{y_t \in \{-1,1\}}{\operatorname{argmax}}\, p_{\mathsf{y}_t|\mathbf{y}_{1:M}}\big(y_t \,\big|\, \boldsymbol{y}_{1:M}; \boldsymbol{X}\big), \tag{4.4}$$

with $\boldsymbol{y}_{1:M} := [y_1, \ldots, y_M]^\mathsf{T}$ and $\boldsymbol{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{M+T}]^\mathsf{T}$. Note that we treat the labels $\{\mathsf{y}_m\}_{m=1}^{M+T}$ as random but the features $\{\boldsymbol{x}_m\}_{m=1}^{M+T}$ as deterministic parameters. The probabilities in (4.4) can be computed via the marginalization

$$p_{\mathsf{y}_t|\mathbf{y}_{1:M}}\big(y_t \,\big|\, \boldsymbol{y}_{1:M}; \boldsymbol{X}\big) = p_{\mathsf{y}_t,\mathbf{y}_{1:M}}\big(y_t, \boldsymbol{y}_{1:M}; \boldsymbol{X}\big) C_\mathbf{y}^{-1} \tag{4.5}$$

$$= C_\mathbf{y}^{-1} \sum_{\boldsymbol{y} \in \mathcal{Y}_t(y_t)} \int p_{\mathbf{y},\mathbf{w}}(\boldsymbol{y}, \boldsymbol{w}; \boldsymbol{X})\, d\boldsymbol{w} \tag{4.6}$$

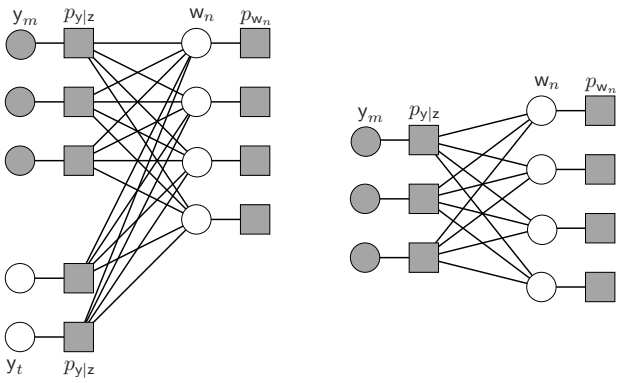
with scaling constant $C_\mathbf{y} := p_{\mathbf{y}_{1:M}}\big(\boldsymbol{y}_{1:M}; \boldsymbol{X}\big)$, label vector $\boldsymbol{y} = [y_1, \ldots, y_{M+T}]^\mathsf{T}$, and constraint set $\mathcal{Y}_t(y) := \{\tilde{\boldsymbol{y}} \in \{-1,1\}^{M+T} \text{ s.t. } [\tilde{\boldsymbol{y}}]_t = y \text{ and } [\tilde{\boldsymbol{y}}]_m = y_m \; \forall m = 1, \ldots, M\}$ 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. The joint pdf in (4.6) factors as

$$p_{\mathbf{y},\mathbf{w}}(\boldsymbol{y}, \boldsymbol{w}; \boldsymbol{X}) = \prod_{m=1}^{M+T} p_{\mathsf{y}_m|\mathsf{z}_m}(y_m \,|\, \boldsymbol{x}_m^\mathsf{T}\boldsymbol{w}) \prod_{n=1}^{N} p_{\mathsf{w}_n}(w_n) \tag{4.7}$$

due to the model (4.1) and assuming a separable prior, i.e.,

$$p_\mathbf{w}(\boldsymbol{w}) = \prod_{n=1}^{N} p_{\mathsf{w}_n}(w_n). \tag{4.8}$$

The factorization (4.7) is illustrated using the *factor graph* in Fig. 4.1a, which connects the various random variables to the pdf factors in which they appear. Although exact computation of the marginal posterior test-label probabilities via (4.6) is computationally intractable due to the high-dimensional summation and integration, the



(a) Full (b) Reduced

Figure 4.1: Factor graph representations of the integrand of (4.7), with white/grey circles denoting unobserved/observed random variables, and rectangles denoting pdf "factors".

factor graph in Fig. 4.1a suggests the use of loopy belief propagation (LBP) [28], and in particular the *sum-product algorithm* (SPA) [55], as a tractable way to approximate these marginal probabilities. Although the SPA guarantees exact marginal posteriors only under non-loopy (i.e., tree-structured graphs), it has proven successful in many applications with loopy graphs, such as turbo decoding [78], computer vision [57], and compressive sensing [25–27].

Because a direct application of the SPA to the factor graph in Fig. 4.1a is itself computationally infeasible in the high-dimensional case of interest, we turn to a recently developed approximation: the sum-product variant of GAMP [27], as specified in Algorithm 1. The GAMP algorithm is specified in Algorithm 1 for a given instantiation of $\boldsymbol{X}$, $p_{\mathsf{y}|\mathsf{z}}$, and $\{p_{\mathsf{w}_n}\}$. There, the expectation and variance in lines 5-6 and 16-17 are taken elementwise w.r.t the GAMP-approximated marginal posterior pdfs

(with superscript k denoting the iteration)

$$q(z_m \,|\, \hat{p}_m^k, \tau_{p_m}^k) = p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|z_m)\, \mathcal{N}(z_m; \hat{p}_m^k, \tau_{p_m}^k)\, C_{\mathsf{z}}^{-1} \tag{4.9}$$

$$q(w_n \,|\, \hat{r}_n^k, \tau_{r_n}^k) = p_{\mathsf{w}_n}(w_n)\, \mathcal{N}(w_n; \hat{r}_n^k, \tau_{r_n}^k)\, C_{\mathsf{w}}^{-1} \tag{4.10}$$

with appropriate normalizations C_{z} and C_{w}, and the vector-vector multiplications and divisions in lines 3, 9, 11, 12, 14, 13, 20 are performed elementwise. Due to space limitations, we refer the interested reader to [27] for an overview and derivation of GAMP, to [30] for rigorous analysis under large i.i.d sub-Gaussian $\boldsymbol{X}$, and to [107,108] for fixed-point and local-convergence analysis under arbitrary $\boldsymbol{X}$.

Applying GAMP to the classification factor graph in Fig. 4.1a and examining the resulting form of lines 5-6 in Algorithm 1, it becomes evident that the test-label nodes $\{\mathsf{y}_t\}_{t=M+1}^{M+T}$ do not affect the GAMP weight estimates $(\hat{\boldsymbol{w}}^k, \boldsymbol{\tau}_w^k)$ and thus the factor graph can effectively be simplified to the form shown in Fig. 4.1b, after which the (approximated) posterior test-label pdfs are computed via

$$p_{\mathsf{y}_t|\mathsf{y}_{1:M}}(y_t|\boldsymbol{y}_{1:M}; \boldsymbol{X}) \approx \int p_{\mathsf{y}_t|\mathsf{z}_t}(y_t|z_t)\, \mathcal{N}(z_t; \hat{z}_t^\infty, \tau_{z,t}^\infty)\, dz_t \tag{4.11}$$

where $\hat{z}_t^\infty$ and $\tau_{z,t}^\infty$ denote the t^{th} element of the GAMP vectors $\hat{\boldsymbol{z}}^k$ and $\boldsymbol{\tau}_z^k$, respectively, at the final iteration "$k = \infty$."

4.2.2 Max-Sum GAMP

An alternate approach to linear classifier design is through the minimization of a regularized loss function, e.g.,

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathbb{R}^N}{\operatorname{argmin}} \sum_{m=1}^{M} f_{\mathsf{z}_m}(\boldsymbol{x}_m^{\mathsf{T}}\boldsymbol{w}) + \sum_{n=1}^{N} f_{\mathsf{w}_n}(w_n), \tag{4.12}$$

Algorithm 1 Generalized Approximate Message Passing

Require: Matrix $\boldsymbol{X}$, priors $p_{\mathsf{w}_n}(\cdot)$, activation functions $p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|\cdot)$, and mode $\in \{\texttt{SumProduct}, \texttt{MaxSum}\}$

Ensure: $k \leftarrow 0$; $\hat{\boldsymbol{s}}^{-1} \leftarrow \mathbf{0}$; $\boldsymbol{S} \leftarrow |\boldsymbol{X}|^2$; $\hat{\boldsymbol{w}}^0 \leftarrow \mathbf{0}$; $\boldsymbol{\tau}_w^0 \leftarrow \mathbf{1}$

repeat
  $\boldsymbol{\tau}_p^k \leftarrow \boldsymbol{S}\boldsymbol{\tau}_w^k$
  $\hat{\boldsymbol{p}}^k \leftarrow \boldsymbol{X}\hat{\boldsymbol{w}}^k - \hat{\boldsymbol{s}}^{k-1}\boldsymbol{\tau}_p^k$
  if `SumProduct` **then**
    $\hat{\boldsymbol{z}}^k \leftarrow \mathrm{E}\{\mathsf{z} \,|\, \hat{\boldsymbol{p}}^k, \boldsymbol{\tau}_p^k\}$
    $\boldsymbol{\tau}_z^k \leftarrow \mathrm{var}\{\mathsf{z} \,|\, \hat{\boldsymbol{p}}^k, \boldsymbol{\tau}_p^k\}$
  else if `MaxSum` **then**
    $\hat{\boldsymbol{z}}^k \leftarrow \mathrm{prox}_{\boldsymbol{\tau}_p^k f_{\mathsf{z}_m}}(\hat{\boldsymbol{p}}^k)$
    $\boldsymbol{\tau}_z^k \leftarrow \boldsymbol{\tau}_p^k \, \mathrm{prox}'_{\boldsymbol{\tau}_p^k f_{\mathsf{z}_m}}(\hat{\boldsymbol{p}}^k)$
  end if
  $\boldsymbol{\tau}_s^k \leftarrow 1/\boldsymbol{\tau}_p^k - \boldsymbol{\tau}_z^k/(\boldsymbol{\tau}_p^k)^2$
  $\hat{\boldsymbol{s}}^k \leftarrow (\hat{\boldsymbol{z}}^k - \hat{\boldsymbol{p}}^k)/\boldsymbol{\tau}_p^k$
  $\boldsymbol{\tau}_r^k \leftarrow 1/(\boldsymbol{S}^{\mathsf{T}}\boldsymbol{\tau}_s^k)$
  $\hat{\boldsymbol{r}}^k \leftarrow \hat{\boldsymbol{w}}^k + \boldsymbol{\tau}_r^k \boldsymbol{X}^{\mathsf{T}}\hat{\boldsymbol{s}}^k$
  if `SumProduct` **then**
    $\hat{\boldsymbol{w}}^{k+1} \leftarrow \mathrm{E}\{\mathsf{w} \,|\, \hat{\boldsymbol{r}}^k, \boldsymbol{\tau}_r^k\}$
    $\boldsymbol{\tau}_w^{k+1} \leftarrow \mathrm{var}\{\mathsf{w} \,|\, \hat{\boldsymbol{r}}^k, \boldsymbol{\tau}_r^k\}$
  else if `MaxSum` **then**
    $\hat{\boldsymbol{w}}^{k+1} \leftarrow \mathrm{prox}_{\boldsymbol{\tau}_r^k f_{\mathsf{w}_n}}(\hat{\boldsymbol{r}}^k)$
    $\boldsymbol{\tau}_w^{k+1} \leftarrow \boldsymbol{\tau}_r^k \, \mathrm{prox}'_{\boldsymbol{\tau}_r^k f_{\mathsf{w}_n}}(\hat{\boldsymbol{r}}^k)$
  end if
  $k \leftarrow k + 1$
until Terminated

where $f_{\mathsf{z}_m}(\cdot)$ are y_m-dependent convex loss functions (e.g., logistic, probit, or hinge based) and where $f_{\mathsf{w}_n}(\cdot)$ are convex regularization terms (e.g., $f_{\mathsf{w}_n}(w) = \lambda w^2$ for ℓ_2 regularization and $f_{\mathsf{w}_n}(w) = \lambda|w|$ for ℓ_1 regularization).

The solution to (4.12) can be recognized as the *maximum a posteriori* (MAP) estimate of random vector w given a separable prior $p_{\mathsf{w}}(\cdot)$ and likelihood corresponding to (4.1), i.e.,

$$p_{\mathsf{y}|\mathsf{w}}(\boldsymbol{y}|\boldsymbol{w};\boldsymbol{X}) = \prod_{m=1}^{M} p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|\boldsymbol{x}_m^{\mathsf{T}}\boldsymbol{w}), \tag{4.13}$$

when $f_{\mathsf{z}_m}(z) = -\log p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|z)$ and $f_{\mathsf{w}_n}(w) = -\log p_{\mathsf{w}_n}(w)$. Importantly, this statistical model is exactly the one yielding the reduced factor graph in Fig. 4.1b.

Similar to how sum-product LBP can be used to compute (approximate) marginal posteriors in loopy graphs, *max-sum* LBP can be used to compute the MAP estimate [109]. Since max-sum LBP is itself intractable for the high-dimensional problems of interest, we turn to the max-sum variant of GAMP [27], which is also specified in Algorithm 1. There, lines 8-9 are to be interpreted as

$$\hat{z}_m^k = \operatorname{prox}_{\tau_{pm}^k f_{\mathsf{z}_m}}(\hat{p}_m^k), \quad m = 1,\dots,M, \tag{4.14}$$

$$\tau_{z_m}^k = \tau_{p_m}^k \operatorname{prox}'_{\tau_{pm}^k f_{\mathsf{z}_m}}(\hat{p}_m^k), \quad m = 1,\dots,M, \tag{4.15}$$

with $(\cdot)'$ and $(\cdot)''$ denoting first and second derivatives and

$$\operatorname{prox}_{\tau f}(v) := \operatorname*{argmin}_{u\in\mathbb{R}} \left[f(u) + \frac{1}{2\tau}(u-v)^2 \right] \tag{4.16}$$

$$\operatorname{prox}'_{\tau f}(v) = \left(1 + \tau f''(\operatorname{prox}_{\tau f}(v))\right)^{-1}, \tag{4.17}$$

and lines 19–20 are to be interpreted similarly. It is known [107] that, for *arbitrary* $\boldsymbol{X}$, the fixed points of GAMP correspond to the critical points of the optimization objective (4.12).

4.2.3 GAMP Summary

In summary, the sum-product and max-sum variants of the GAMP algorithm provide tractable methods of approximating the posterior test-label probabilities $\{p_{\mathsf{y}_t|\mathsf{y}_{1:M}}(y_t|\boldsymbol{y}_{1:M})\}_{t=T+1}^{M+T}$ and finding the MAP weight vector $\hat{\boldsymbol{w}} = \operatorname{argmax}_{\boldsymbol{w}} p_{\mathsf{w}|\mathsf{y}_{1:M}}(\boldsymbol{w}|\boldsymbol{y}_{1:M})$, respectively, under the label-generation model (4.13) [equivalently, (4.1)] and the separable weight-vector prior (4.8), assuming that the distributions $p_{\mathsf{y}|\mathsf{z}}$ and $\{p_{\mathsf{w}_n}\}$ are known and facilitate tractable scalar-nonlinear update steps 5-6, 8-9, 16-17, and 19-20. In Section 4.4, we discuss the implementation of these update steps for several popular activation functions; and in Section 4.5, we discuss how the parameters of $p_{\mathsf{y}_m|\mathsf{z}_m}$ and p_{w_n} can be learned online.

4.3 Predicting Misclassification Rate via State Evolution

As mentioned earlier, the behavior of GAMP in the large-system limit (i.e., $M, N \to \infty$ with fixed ratio $\delta = \frac{M}{N}$) under i.i.d sub-Gaussian $\boldsymbol{X}$ is characterized by a scalar state evolution [27, 30]. We now describe how this state evolution can be used to characterize the test-error rate of the linear-classification GAMP algorithms described in Section 4.2.

The GAMP state evolution characterizes average GAMP performance over an ensemble of (infinitely sized) problems, each associated with one realization $(\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{w})$ of the random triple $(\mathsf{y}, \mathsf{X}, \mathsf{w})$. Recall that, for a given problem realization $(\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{w})$, the GAMP iterations in Algorithm 1 yields the sequence of estimates $\{\hat{\boldsymbol{w}}^k\}_{k=1}^{\infty}$ of the true weight vector $\boldsymbol{w}$. Then, according to the state evolution, $p_{\mathsf{w},\hat{\mathsf{w}}^k}(\boldsymbol{w}, \hat{\boldsymbol{w}}^k) \sim \prod_n p_{\mathsf{w}_n,\hat{\mathsf{w}}_n^k}(w_n, \hat{w}_n^k)$ and the first two moments of the joint pdf $p_{\mathsf{w}_n,\hat{\mathsf{w}}_n^k}$ can be computed using [27, Algorithm 3].

Suppose that the (y, X) above represent training examples associated with a true

weight vector $\mathbf{w}$, and that $(\mathsf{y}, \mathbf{x})$ represents a test pair also associated with the same $\mathbf{w}$ and with $\mathbf{x}$ having i.i.d elements distributed identically to those of $\mathbf{X}$ (with, say, variance $\frac{1}{M}$). The true and iteration-k-estimated test scores are then $\mathsf{z} \triangleq \mathbf{x}^\mathsf{T}\mathbf{w}$ and $\hat{\mathsf{z}}^k \triangleq \mathbf{x}^\mathsf{T}\widehat{\mathbf{w}}^k$, respectively. The corresponding test-error rate[5] $\mathcal{E}^k := \Pr\{\mathsf{y} \neq \operatorname{sgn}(\hat{\mathsf{z}}^k)\}$ can be computed as follows. Letting $I_{\{\cdot\}}$ denote an indicator function that assumes the value 1 when its Boolean argument is true and the value 0 otherwise, we have

$$\begin{aligned}
\mathcal{E}^k &= \mathrm{E}\big\{I_{\{\mathsf{y}\neq\operatorname{sgn}(\hat{\mathsf{z}}^k)\}}\big\} && (4.18)\\
&= \sum_{y\in\{-1,1\}} \int I_{\{y\neq\operatorname{sgn}(\hat{z}^k)\}} \int p_{\mathsf{y},\hat{\mathsf{z}}^k,\mathsf{z}}(y,\hat{z}^k,z)\,dz\,d\hat{z}^k && (4.19)\\
&= \sum_{y\in\{-1,1\}} \iint I_{\{y\neq\operatorname{sgn}(\hat{z}^k)\}} p_{\mathsf{y}|\mathsf{z}}(y|z) p_{\mathsf{z},\hat{\mathsf{z}}^k}(z,\hat{z}^k)\,dz\,d\hat{z}^k. && (4.20)
\end{aligned}$$

Furthermore, from the definitions of $(\mathsf{z}, \hat{\mathsf{z}}^k)$ and the bivariate central limit theorem, we have that

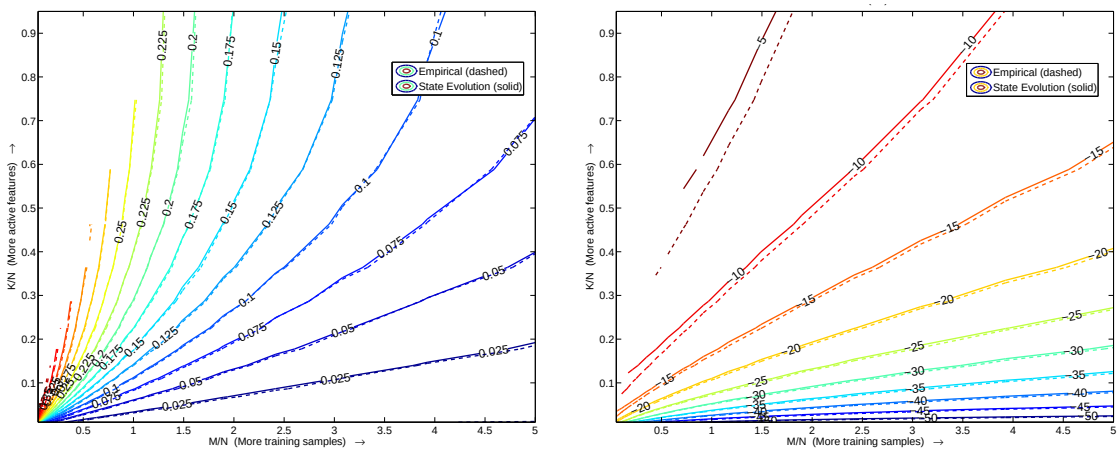
$$\begin{bmatrix}\mathsf{z}\\ \hat{\mathsf{z}}^k\end{bmatrix} \xrightarrow{d} \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_z^k) = \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\Sigma_{11}^k & \Sigma_{12}^k\\ \Sigma_{21}^k & \Sigma_{22}^k\end{bmatrix}\right), \tag{4.21}$$

where $\xrightarrow{d}$ indicates convergence in distribution. In Appendix E.1, it is shown that the above matrix components are

$$\begin{aligned}
\Sigma_{11}^k = \tau_x \sum_{n=1}^N [\operatorname{var}\{\mathsf{w}_n\} + \mathrm{E}[\mathsf{w}_n]^2], \quad \Sigma_{22}^k = \tau_x \sum_{n=1}^N [\operatorname{var}\{\widehat{\mathsf{w}}_n^k\} + \mathrm{E}[\widehat{\mathsf{w}}_n^k]^2],\\
\Sigma_{12}^k = \Sigma_{21}^k = \mathrm{E}[\mathsf{z}\hat{\mathsf{z}}^k] = \tau_x \sum_{n=1}^N [\operatorname{cov}\{\mathsf{w}_n, \widehat{\mathsf{w}}_n^k\} + \mathrm{E}[\mathsf{w}_n]\mathrm{E}[\widehat{\mathsf{w}}_n^k]]. \qquad (4.22)
\end{aligned}$$

for label-to-feature ratio δ. As described earlier, the above moments can be computed using [27, Algorithm 3]. The integral in (4.20) can then be computed (numerically if

[5] For simplicity we assume a decision rule of the form $\hat{\mathsf{y}}^k = \operatorname{sgn}(\hat{\mathsf{z}}^k)$, although other decision rules can be accommodated in our analysis.



(a) Test-Error Rate

(b) Weight-Vector MSE (dB)

Figure 4.2: Test-error rate (a) and weight-vector MSE (b), versus training-to-feature ratio M/N and discriminative-feature ratio K/N, calculated using empirical averaging (dashed) and state-evolution prediction (solid), assuming i.i.d Bernoulli-Gaussian weight vectors and a probit activation function.

needed) for a given activation function $p_{\mathsf{y}|\mathsf{z}}$, yielding an estimate of GAMP's test-error rate at the k^{th} iteration.

To validate the accuracy of the above asymptotic analysis, we conducted a Monte-Carlo experiment with data synthetically generated in accordance with the assumed model. In particular, for each of 1000 problem realizations, a true weight vector $\boldsymbol{w} \in \mathbb{R}^N$ was drawn i.i.d zero-mean Bernoulli-Gaussian and a feature matrix $\boldsymbol{X}$ was drawn i.i.d Gaussian, yielding true scores $\boldsymbol{z} = \boldsymbol{X}\boldsymbol{w}$, from which the true labels $\boldsymbol{y}$ were randomly drawn using a probit activation function $p_{\mathsf{y}|\mathsf{z}}$. A GAMP weight-vector estimate $\hat{\boldsymbol{w}}^\infty$ was then computed using the training data $(\boldsymbol{y}_{1:M}, \boldsymbol{X}_{1:M})$, from which the test-label estimates $\{\hat{y}_t^\infty\}_{t=M+1}^{M+T}$ with $\hat{y}_t^\infty = \mathrm{sgn}(\boldsymbol{x}_t^\mathsf{T}\hat{\boldsymbol{w}}^\infty)$ were computed and compared to the true test-labels in order to calculate the test-error rate for that realization. Figure 4.2a plots the Monte-Carlo averaged empirical test-error rates (dashed) and state-evolution predicted rates (solid) as level curves over different combinations of training ratio $\frac{M}{N}$ and discriminative-feature ratio $\frac{K}{N}$, where $K = \|\boldsymbol{w}\|_0$ and $N = 1024$.

Similarly, Fig. 4.2b plots average empirical mean-squared error (MSE) versus state-evolution predicted MSE, where $\text{MSE} = \frac{1}{N}\text{E}\{\|\hat{\boldsymbol{w}}^{\infty} - \boldsymbol{w}\|_2^2\}$.

In both Fig. 4.2a and Fig. 4.2b, the training-to-feature ratio $\frac{M}{N}$ increases from left to right, and the discriminative-feature ratio $\frac{K}{N}$ increases from bottom to top. The region to the upper-left of the dash-dotted black line contains ill-posed problems (where the number of discriminative features K exceeds the number of training samples M) for which data was not collected. The remainders of Fig. 4.2a and Fig. 4.2b show very close agreement between empirical averages and state-evolution predictions.

4.4 GAMP for Classification

Section 4.2 gave a high-level description of how the GAMP iterations in Algorithm 1 can be applied to binary linear classification and feature selection. In this section, we detail the nonlinear steps used to compute $(\hat{z}, \tau_z)$ and $(\hat{x}, \tau_x)$ in lines 5-6, 8-9, 16-17, and 19-20 of Algorithm 1. For sum-product GAMP, we recall that the mean and variance computations in lines 5-6 and 16-17 are computed based on the pdfs in (4.9) and (4.10), respectively, and for max-sum GAMP the prox steps in 8-9 are computed using equations (4.14)-(4.15) and those in 19-20 are computed similarly.

4.4.1 Logistic Classification Model

Arguably the most popular activation function for binary linear classification is the logistic sigmoid [67, §4.3.2], [110]:

$$p_{\mathsf{y}|\mathsf{z}}(y|z;\alpha) = \frac{1}{1+\exp(-y\alpha z)}, \quad y \in \{-1, 1\} \tag{4.23}$$

where $\alpha > 0$ controls the steepness of the transition.

For logistic sum-product GAMP, we propose to compute the mean and variance

$(\hat{z}, \tau_z)$ of the marginal posterior approximation (4.9) using the variational approach in Algorithm 2, a derivation of which is provided in Appendix E.2. We note that Algorithm 2 is reminiscent of the one presented in [67, §10.6], but is more general in that it handles $\alpha \neq 1$.

For logistic max-sum GAMP, $\hat{z}$ from (4.14) solves the scalar minimization problem (4.16) with $f(u) = -\log p_{\mathsf{y}|\mathsf{z}}(y|u;\alpha)$ from (4.23), which is convex. To find this $\hat{z}$, we use bisection search to locate the root of $\frac{d}{du}[f(u) + \frac{1}{2\tau}(u-v)^2]$. The max-sum τ_z from (4.15) can then be computed in closed form using $\hat{z}$ and $f''(\cdot)$ via (4.17). Note that, unlike the classical ML-based approach to logistic regression (e.g., [67, §4.3.3]), GAMP performs only scalar minimizations and thus does not need to construct or invert a Hessian matrix.

Algorithm 2 A Variational Approach to Logistic Activation Functions for Sum-Product GAMP

Require: Class label $y \in \{-1, 1\}$, logistic scale α, and GAMP-computed parameters $\hat{p}$ and τ_p (see (4.9))

Ensure: $\xi \leftarrow \sqrt{\tau_p + |\hat{p}|^2}$

1: **repeat**
2: $\quad \sigma \leftarrow \big(1 + \exp(-\alpha\xi)\big)^{-1}$
3: $\quad \lambda \leftarrow \frac{\alpha}{2\xi}(\sigma - \frac{1}{2})$
4: $\quad \tau_z \leftarrow \tau_p(1 + 2\tau_p\lambda)^{-1}$
5: $\quad \hat{z} \leftarrow \tau_z(\hat{p}/\tau_p + \alpha y/2)$
6: $\quad \xi \leftarrow \sqrt{\tau_z + |\hat{z}|^2}$
7: **until** Terminated
8: **return** $\hat{z}$, τ_z

Quantity	Value
c	$\dfrac{\hat{p}}{\sqrt{v+\tau_p}}$
$\hat{z}$	$\hat{p} + \dfrac{y\tau_p\phi(c)}{\Phi(yc)\sqrt{v+\tau_p}}$
τ_z	$\tau_p - \dfrac{\tau_p^2\phi(c)}{\Phi(yc)(v+\tau_p)}\left(yc + \dfrac{\phi(c)}{\Phi(c)}\right)$

Table 4.1: Sum-product GAMP computations for probit activation function.

4.4.2 Probit Classification Model

Another popular activation function is the probit [67, §4.3.5]:

$$p_{\mathsf{y}|\mathsf{z}}(1|z;v) \;=\; \int_{-\infty}^{z} \mathcal{N}(\tau;0,v)d\tau = \Phi\Big(\frac{z}{\sqrt{v}}\Big) \tag{4.24}$$

where $p_{\mathsf{y}|\mathsf{z}}(-1|z) = 1 - p_{\mathsf{y}|\mathsf{z}}(1|z) = \Phi(-\frac{z}{\sqrt{v}})$ and where $v > 0$ controls the steepness of the sigmoid.

Unlike the logistic case, the probit case leads to closed-form sum-product GAMP computations. In particular, the density (4.9) corresponds to the posterior pdf of a random variable z with prior $\mathcal{N}(\hat{p},\tau_p)$ from an observation $\mathsf{y} = y$ measured under the likelihood model (4.24). A derivation in [111, §3.9] provides the necessary expressions for these moments when $y = 1$, and a similar exercise tackles the $y = -1$ case. For completeness, the sum-product computations are summarized in Table 4.1. Max-sum GAMP computation of $(\hat{z},\tau_z)$ can be performed using a bisection search akin to that described in Section 4.4.1.

4.4.3 Hinge Loss Classification Model

In addition to the two preceding generalized linear models, another commonly used choice for binary classification is a maximum-margin classifier—a classifier that (in the linearly separable case) seeks a hyperplane that maximizes the perpendicular distance to the closest point from either class (i.e., the margin). The popular *support vector machine* (SVM) [112, 113] is one such classifier.

As described in [67, §7.1], the SVM solves the following optimization problem:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \sum_{m=1}^{M} \Theta_H(y_m, z_m) + \lambda \|\boldsymbol{w}\|_2^2, \\ \text{s.t.} \quad & z_m = \boldsymbol{x}_m^{\mathsf{T}} \boldsymbol{w} + b \quad \forall m, \end{aligned} \tag{4.25}$$

where $\Theta_H(y, z)$ is the *hinge loss* function,

$$\Theta_H(y, z) \triangleq \max(0, 1 - yz). \tag{4.26}$$

The hinge loss function (4.26) can be interpreted as the (negative) log-likelihood corresponding to the following (improper) pdf:

$$p_{\mathsf{y}|\mathsf{z}}(y|z) \quad \propto \quad \frac{1}{\exp(\max(0, 1 - yz))}. \tag{4.27}$$

Despite being an improper likelihood, (4.27) can be paired with a variety of prior distributions, $p(\boldsymbol{w})$, in order to yield different flavors of maximum-margin classifiers (e.g., a Gaussian prior with variance $1/2\lambda$ yields the SVM). As with both logistic and probit regression, the max-sum GAMP updates can be efficiently computed using simple numerical optimization techniques. MMSE estimation via sum-product GAMP can be accomplished in closed-form via an update procedure which is described in Appendix E.3.

4.4.4 A Method to Robustify Activation Functions

In some applications, a fraction $\gamma \in (0,1)$ of the training labels are known[6] to be corrupted, or at least highly atypical under a given activation model $p^*_{\mathsf{y}|\mathsf{z}}(y|z)$. As a robust alternative to $p^*_{\mathsf{y}|\mathsf{z}}(y|z)$, Opper and Winther [114] proposed to use

$$p_{\mathsf{y}|\mathsf{z}}(y|z;\gamma) = (1-\gamma)p^*_{\mathsf{y}|\mathsf{z}}(y|z) + \gamma p^*_{\mathsf{y}|\mathsf{z}}(-y|z) \tag{4.28}$$

$$= \gamma + (1-2\gamma)p^*_{\mathsf{y}|\mathsf{z}}(y|z). \tag{4.29}$$

We now describe how the GAMP nonlinear steps for an arbitrary $p^*_{\mathsf{y}|\mathsf{z}}$ can be used to compute the GAMP nonlinear steps for a robust $p_{\mathsf{y}|\mathsf{z}}$ of the form in (4.29).

In the sum-product case, knowledge of the non-robust quantities $\hat{z}^* \triangleq \frac{1}{C^*_y}\int_z z\, p^*_{\mathsf{y}|\mathsf{z}}(y|z)\mathcal{N}(z;\hat{p},\tau_p)$, $\tau^*_z \triangleq \frac{1}{C^*_y}\int_z (z-\hat{z}^*)^2\, p^*_{\mathsf{y}|\mathsf{z}}(y|z)\mathcal{N}(z;\hat{p},\tau_p)$, and $C^*_y \triangleq \int_z p^*_{\mathsf{y}|\mathsf{z}}(y|z)\mathcal{N}(z;\hat{p},\tau_p)$ is sufficient for computing the robust sum-product quantities $(\hat{z},\tau_z)$, as summarized in Table 4.2. (See Appendix E.4 for details.)

In the max-sum case, computing $\hat{z}$ in (4.14) involves solving the scalar minimization problem in (4.16) with $f(u) = -\log p_{\mathsf{y}|\mathsf{z}}(y|u;\gamma) = -\log[\gamma + (1-2\gamma)p^*_{\mathsf{y}|\mathsf{z}}(y|u)]$. As before, we use a bisection search to find $\hat{z}$ and then we use $f''(\hat{z})$ to compute τ_z via (4.17).

4.4.5 Weight Vector Priors

We now discuss the nonlinear steps used to compute $(\hat{w},\tau_w)$, i.e., lines 16-17 and 19-20 of Algorithm 1. These steps are, in fact, identical to those used to compute $(\hat{z},\tau_z)$ except that the prior $p_{\mathsf{w}_n}(\cdot)$ is used in place of the activation function $p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|\cdot)$. For linear classification and feature selection in the $N \gg M$ regime, it is customary

[6] A method to learn an unknown γ will be proposed in Section 4.5.

Quantity	Value
C_y	$\dfrac{\gamma}{\gamma + (1-2\gamma)C_y^*}$
$\hat{z}$	$C_y\hat{p} + (1-C_y)\hat{z}^*$
τ_z	$C_y(\tau_p + \hat{p}^2) + (1-C_y)(\tau_z^* + (\hat{z}^*)^2) - \hat{z}^2$

Table 4.2: Sum-product GAMP computations for a robustified activation function. See text for definitions of C_y^*, $\hat{z}^*$, and τ_z^*.

to choose a prior $p_{\mathsf{w}_n}(\cdot)$ that leads to sparse (or approximately sparse) weight vectors $\boldsymbol{w}$, as discussed below.

For sum-product GAMP, this can be accomplished by choosing a Bernoulli-$\tilde{p}$ prior, i.e.,

$$p_{\mathsf{w}_n}(w) = (1-\pi_n)\delta(w) + \pi_n \tilde{p}_{\mathsf{w}_n}(w), \tag{4.30}$$

where $\delta(\cdot)$ is the Dirac delta function, $\pi_n \in [0,1]$ is the prior[7] probability that $\mathsf{w}_n = 0$, and $\tilde{p}_{\mathsf{w}_n}(\cdot)$ is the pdf of a non-zero w_n. While Bernoulli-Gaussian [53] and Bernoulli-Gaussian-mixture [106] are common choices, Section 4.6 suggests that Bernoulli-Laplacian also performs well.

In the max-sum case, the GAMP nonlinear outputs $(\hat{w}, \tau_w)$ are computed via

$$\hat{w} = \text{prox}_{\tau_r f_{\mathsf{w}_n}}(\hat{r}) \tag{4.31}$$

$$\tau_w = \tau_r \, \text{prox}'_{\tau_r f_{\mathsf{w}_n}}(\hat{r}) \tag{4.32}$$

for a suitably chosen regularizer $f_{\mathsf{w}_n}(w)$. Common examples include $f_{\mathsf{w}_n}(w) = \lambda_1|w|$ for ℓ_1 regularization [25], $f_{\mathsf{w}_n}(w) = \lambda_2 w^2$ for ℓ_2 regularization [27], and $f_{\mathsf{w}_n}(w) = \lambda_1|w| + \lambda_2 w^2$ for the "elastic net" [62]. As described in Section 4.2.2, any regularizer

[7] In Section 4.5 we describe how a common $\pi = \pi_n \; \forall n$ can be learned.

	Quantity	Value
SPG	$\hat{w}$	$\big(\underline{C}\underline{\mu}+\bar{C}\bar{\mu}\big)/\big(\underline{C}+\bar{C}\big)$
	τ_w	$\big(\underline{C}(\underline{v}+\underline{\mu}^2)+\bar{C}(\bar{v}+\bar{\mu}^2)\big)/\big(\underline{C}+\bar{C}\big)-\hat{w}^2$
MSG	$\hat{w}$	$\operatorname{sgn}(\sigma\ddot{r})\max(\lvert\sigma\ddot{r}\rvert-\lambda_1\sigma^2,0)$
	τ_w	$\sigma^2\cdot\mathrm{I}_{\{\hat{w}\neq 0\}}$

Table 4.3: Sum-product GAMP (SPG) and max-sum GAMP (MSG) computations for the elastic-net regularizer $f_{\mathsf{w}_n}(w)=\lambda_1|w|+\lambda_2 w^2$, which includes ℓ_1 or Laplacian-prior (via $\lambda_2=0$) and ℓ_2 or Gaussian-prior (via $\lambda_1=0$) as special cases. See Table 4.4 for definitions of $\underline{C}$, $\bar{C}$, $\underline{\mu}$, $\bar{\mu}$, etc.

$\sigma \triangleq \sqrt{\tau_r/(2\lambda_2\tau_r+1)}$	$\ddot{r} \triangleq \hat{r}/(\sigma(2\lambda_2\tau_r+1))$
$\underline{r} \triangleq \ddot{r}+\lambda_1\sigma$	$\bar{r} \triangleq \ddot{r}-\lambda_1\sigma$
$\underline{C} \triangleq \frac{\lambda_1}{2}\exp\left(\frac{\underline{r}^2-\ddot{r}^2}{2}\right)\Phi(-\underline{r})$	$\bar{C} \triangleq \frac{\lambda_1}{2}\exp\left(\frac{\bar{r}^2-\ddot{r}^2}{2}\right)\Phi(\bar{r})$
$\underline{\mu} \triangleq \sigma\underline{r}-\sigma\phi(-\underline{r})/\Phi(-\underline{r})$	$\bar{\mu} \triangleq \sigma\bar{r}+\sigma\phi(\bar{r})/\Phi(\bar{r})$
$\underline{v} \triangleq \sigma^2\left[1-\frac{\phi(\underline{r})}{\Phi(\underline{r})}\left(\frac{\phi(\underline{r})}{\Phi(\underline{r})}-\underline{r}\right)\right]$	$\bar{v} \triangleq \sigma^2\left[1-\frac{\phi(\bar{r})}{\Phi(\bar{r})}\left(\frac{\phi(\bar{r})}{\Phi(\bar{r})}+\bar{r}\right)\right]$

Table 4.4: Definitions of elastic-net quantities used in Table 4.3.

f_{w_n} can be interpreted as a (possibly improper) prior pdf $p_{\mathsf{w}_n}(w)\propto\exp(-f_{\mathsf{w}_n}(w))$. Thus, ℓ_1 regularization corresponds to a Laplacian prior, ℓ_2 to a Gaussian prior, and the elastic net to a product of Laplacian and Gaussian pdfs.

In Table 4.6, we give the sum-product and max-sum computations for the prior corresponding to the elastic net, which includes both Laplacian (i.e., ℓ_1) and Gaussian (i.e., ℓ_2) as special cases; a full derivation can be found in Appendix F.2. For the Bernoulli-Laplacian case, these results can be combined with the Bernoulli-$\tilde{p}$ extension in Table 4.6.

Name	$p_{\mathsf{y}\|\mathsf{z}}(y\|z)$ Description	Sum-Product	Max-Sum
Logistic	$\propto (1+\exp(-\alpha yz))^{-1}$	VI	RF
Probit	$\Phi\left(\frac{yz}{v}\right)$	CF	RF
Hinge Loss	$\propto \exp(-\max(0, 1-yz))$	CF	RF
Robust-p^*	$\gamma + (1-2\gamma)p^*_{\mathsf{y}\|\mathsf{z}}(y\|z)$	CF	RF

Table 4.5: Activity-functions and their GAMPmatlab sum-product and max-sum implementation method: CF = closed form, VI = variational inference, RF = root-finding.

Name	$p_{\mathsf{w}_n}(w)$ Description	Sum-Product	Max-Sum
Gaussian	$\mathcal{N}(w;\mu,\sigma^2)$	CF	CF
GM	$\sum_l \omega_l \mathcal{N}(w;\mu_l,\sigma_l^2)$	CF	NI
Laplacian	$\propto \exp(-\lambda\|w\|)$	CF	CF
Elastic Net	$\propto \exp(-\lambda_1\|w\| - \lambda_2 w^2)$	CF	CF
Bernoulli-$\tilde{p}$	$(1-\pi_n)\delta(w) + \pi_n \tilde{p}_{\mathsf{w}_n}(w)$	CF	NA

Table 4.6: Weight-coefficient priors and their GAMPmatlab sum-product and max-sum implementation method: CF = closed form, NI = not implemented, NA = not applicable.

4.4.6 The GAMP Software Suite

The GAMP iterations from Algorithm 1, including the nonlinear steps discussed in this section, have been implemented in the open-source "GAMPmatlab" software suite.[8] For convenience, the existing activation-function implementations are summarized in Table 4.5 and relevant weight-prior implementations appear in Table 4.6.

4.5 Automatic Parameter Tuning

The activation functions and weight-vector priors described in Section 4.4 depend on modeling parameters that, in practice, must be tuned. For example, the logistic

[8]The latest source code can be obtained through the GAMPmatlab SourceForge Subversion repository at http://sourceforge.net/projects/gampmatlab/.

sigmoid (4.23) depends on α; the probit depends on v; ℓ_1 regularization depends on λ; and the Bernoulli-Gaussian-mixture prior depends on π and $\{\omega_l, \mu_l, \sigma_l^2\}_{l=1}^L$, where ω_l parameterizes the weight, μ_l the mean, and σ_l^2 the variance of the lth mixture component. Although cross-validation (CV) is the customary approach to tuning parameters such as these, it suffers from two major drawbacks: First, 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 cross-validation tuning of P parameters using G hypothesized values of each requires the training and evaluation of KG^P classifiers. Second, leaving out a portion of the training data for CV can degrade classification performance, especially in the example-starved regime where $M \ll N$ (see, e.g., [115]).

As an alternative to CV, we consider *online learning* of the unknown model parameters $\boldsymbol{\theta}$, employing two complementary strategies. The first strategy relies on a traditional expectation-maximization (EM) approach [65] to parameter tuning, while the second strategy leverages an alternative interpretation of sum-product GAMP as a Bethe free entropy minimization algorithm [107].

4.5.1 Traditional EM Parameter Tuning

Our first approach to online learning employs a methodology described in [106, 116]. Here, the goal is to compute the maximum-likelihood estimate $\hat{\boldsymbol{\theta}}_{\mathsf{ML}} = \operatorname{argmax}_{\boldsymbol{\theta}} p_{\mathbf{y}}(\boldsymbol{y}; \boldsymbol{\theta})$, where our data model implies a likelihood function of the form

$$p_{\mathbf{y}}(\boldsymbol{y}; \boldsymbol{\theta}) = \int_{\boldsymbol{w}} \prod_m p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|\boldsymbol{x}^{\mathsf{T}}\boldsymbol{w}; \boldsymbol{\theta}) \prod_n p_{\mathsf{w}_n}(w_n; \boldsymbol{\theta}). \tag{4.33}$$

Because it is computationally infeasible to evaluate and/or maximize (4.33) directly, we apply the expectation-maximization (EM) algorithm [65]. For EM, we treat $\mathbf{w}$ as

the "hidden" data, giving the iteration-j EM update

$$\boldsymbol{\theta}^j = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \, \mathrm{E}_{\mathsf{w}|\mathsf{y}}\big\{ \log \, p_{\mathsf{y},\mathsf{w}}(\boldsymbol{y}, \mathbf{w}; \boldsymbol{\theta}) \,\big|\, \boldsymbol{y}; \boldsymbol{\theta}^{j-1} \big\} \tag{4.34}$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_m \mathrm{E}_{\mathsf{z}_m|\mathsf{y}}\big\{ \log p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|\mathsf{z}_m; \boldsymbol{\theta}) \,\big|\, \boldsymbol{y}; \boldsymbol{\theta}^{j-1} \big\}$$

$$+ \sum_n \mathrm{E}_{\mathsf{w}_n|\mathsf{y}}\big\{ \log p_{\mathsf{w}_n}(\mathsf{w}_n; \boldsymbol{\theta}) \,\big|\, \boldsymbol{y}; \boldsymbol{\theta}^{j-1} \big\}. \tag{4.35}$$

Furthermore, to evaluate the conditional expectations in (4.35), GAMP's posterior approximations from (4.9)-(4.10) are used. It was shown in [117] that, in the large-system limit, the estimates generated by this procedure are asymptotically consistent (as $j \to \infty$ and under certain identifiability conditions). Moreover, it was shown in [106, 116] that, for various priors and likelihoods of interest in compressive sensing (e.g., AWGN likelihood, Bernoulli-Gaussian-Mixture priors, ℓ_1 regularization), the quantities needed from the expectation in (4.35) are implicitly computed by GAMP, making this approach computationally attractive. However, because this EM procedure runs GAMP several times, once for each EM iteration (although not necessarily to convergence), the total runtime may be increased relative to that of GAMP without EM.

In this chapter, we propose EM-based learning of the activation-function parameters, i.e., α in the logistic model (4.23), v in the probit model (4.24), and γ in the robust model (4.29). Starting with α, we find that a closed-form expression for the value maximizing (4.35) remains out of reach, due to the form of the logistic model (4.23). So, we apply the same variational lower bound used for Algorithm 2, and find that the lower-bound maximizing value of α obeys (see Appendix E.6)

$$0 = \sum_m \tfrac{1}{2}(\hat{z}_m y_m - \xi_m) + \frac{\xi_m}{1 + \exp(\alpha \xi_m)}, \tag{4.36}$$

where ξ_m is the variational parameter being used to optimize the lower-bound and $\hat{z}_m \triangleq \mathrm{E}_{\mathsf{z}_m|\mathsf{y}}$ is computed in Algorithm 2. We then solve for α using Newton's method.

To tune the probit parameter, v, we zero the derivative of (4.35) w.r.t v to obtain

$$0 = \sum_m \mathrm{E}_{\mathsf{z}_m|\mathsf{y}}\big\{\tfrac{\partial}{\partial v}\log p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|\mathsf{z}_m;v)\,\big|\,\boldsymbol{y};v^{j-1}\big\} \tag{4.37}$$

$$= \sum_m \mathrm{E}_{\mathsf{z}_m|\mathsf{y}}\big\{\tfrac{-\ddot{c}_m(v)}{v}\phi(\ddot{c}_m(v))\Phi(\ddot{c}_m(v))^{-1}\big|\boldsymbol{y};v^{j-1}\big\}, \tag{4.38}$$

where $\ddot{c}_m(v) \triangleq (y_m z_m)/v$. Since (4.38) is not easily solved in closed-form for v, we numerically evaluate the expectation, and apply an iterative root-finding procedure to locate the zero-crossing.

To learn γ, we include the corruption indicators $\boldsymbol{\beta}\in\{0,1\}^M$ in the EM-algorithm's hidden data (i.e., $\beta_m=0$ indicates that y_m was corrupt and $\beta_m=1$ that it was not), where an i.i.d assumption on the corruption mechanism implies the prior $p(\boldsymbol{\beta};\gamma) = \prod_{m=1}^M \gamma^{1-\beta_m}(1-\gamma)^{\beta_m}$. In this case, it is shown in Appendix E.5 that the update of the γ parameter reduces to

$$\gamma^j = \operatorname*{argmax}_{\gamma\in[0,1]} \sum_{m=1}^M \mathrm{E}_{\beta_m|\mathsf{y}}\big[\log p(\beta_m;\gamma)\,\big|\,\boldsymbol{y};\boldsymbol{\theta}^{j-1}\big] \tag{4.39}$$

$$= \frac{1}{M}\sum_{m=1}^M p(\beta_m=0\,|\,\boldsymbol{y};\boldsymbol{\theta}^{j-1}), \tag{4.40}$$

where (4.40) leveraged $\mathrm{E}[\beta_m|\boldsymbol{y};\boldsymbol{\theta}^{j-1}] = 1 - p(\beta_m=0|\boldsymbol{y};\boldsymbol{\theta}^{j-1})$. Moreover, $p(\beta_m = 0|\boldsymbol{y};\boldsymbol{\theta}^{j-1})$ is easily computed using quantities returned by sum-product GAMP.

4.5.2 Tuning via Bethe Free Entropy Minimization

While the EM learning strategy proposed above is oftentimes effective, in certain low SNR regimes it can be outperformed by an alternative strategy that leverages a unique interpretation of the sum-product GAMP algorithm, distinct from the interpretation

presented in Section 4.2.1. As proven in [107] and later intrepreted in the context of Bethe free entropy in [118], the fixed points of the sum-product GAMP algorithm coincide with critical points of the optimization problem

$$(f_{\mathsf{w}}, f_{\mathsf{z}}) = \underset{b_{\mathsf{w}}, b_{\mathsf{z}}}{\operatorname{argmin}} J(b_{\mathsf{w}}, b_{\mathsf{z}}) \text{ such that } \mathrm{E}[\mathsf{w}|b_{\mathsf{w}}] = \boldsymbol{X}\mathrm{E}[\mathsf{z}|b_{\mathsf{z}}] \tag{4.41}$$

for the cost function

$$J(b_{\mathsf{w}}, b_{\mathsf{z}}) \triangleq D\big(b_{\mathsf{w}} \| p_{\mathsf{w}}\big) + D\big(b_{\mathsf{z}} \| p_{\mathsf{y}|\mathsf{z}} Z^{-1}\big) + H_{\text{gauss}}\big(b_{\mathsf{z}}, |\boldsymbol{X}|^2 \text{var}\{\mathsf{w}|b_{\mathsf{w}}\}\big), \tag{4.42}$$

where the optimization quantities b_{w} and b_{z} are separable pdfs over w and z respectively, $Z^{-1} = \int_{\boldsymbol{z}} p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y}|\boldsymbol{z})$ is the scaling factor that renders $p_{\mathsf{y}|\mathsf{z}}(\boldsymbol{y}|\boldsymbol{z})Z^{-1}$ a valid pdf in $\boldsymbol{z}$, $D(\cdot\|\cdot)$ denotes Kullback-Leibler divergence,

$$H_{\text{gauss}}\big(b_{\mathsf{z}}, \boldsymbol{\tau}_p\big) \triangleq \frac{1}{2}\sum_{m=1}^{M}\left(\frac{\text{var}\{\mathsf{z}_m|b_{\mathsf{z}_m}\}}{\tau_{p_m}} + \ln 2\pi\tau_{p_m}\right) \tag{4.43}$$

is an upper bound on the entropy $H(b_{\mathsf{z}})$ that becomes tight when b_{z} is separable Gaussian with variances in $\boldsymbol{\tau}_p$, and $|\boldsymbol{X}|^2$ is a matrix whose entries are elementwise magnitude squared.

Recalling (4.9) and (4.10), the posterior approximations provided by GAMP have the form

$$f_{\mathsf{w}}(\boldsymbol{w}) = \prod_{n=1}^{N} q(w_n \,|\, \hat{r}_n^k, \tau_{r_n}^k) \tag{4.44}$$

$$f_{\mathsf{z}}(\boldsymbol{z}) = \prod_{m=1}^{M} q(z_m \,|\, \hat{p}_m^k, \tau_{p_m}^k), \tag{4.45}$$

for iteration k-dependent quantities $\hat{r}_n^k, \tau_{r_n}^k, \hat{p}_m^k, \tau_{p_m}^k$. When the latter quantities have

converged, the resulting f_{w} and f_{z} are critical points of the optimization problem (4.41).

Our alternative approach to parameter tuning of output channel ($p_{\mathsf{y}|\mathsf{z}}$-related) parameters (discussed in [119] in the context of the simpler AMP algorithm) leverages the fact that, in the large-system limit, the log-likelihood $\ln p(\boldsymbol{y};\boldsymbol{\theta})$ equals the Bethe free entropy $J(f_{\mathsf{w}}, f_{\mathsf{z}})$ for convergent $(f_{\mathsf{w}}, f_{\mathsf{z}})$. It can be shown that the latter manifests as,

$$J(f_{\mathsf{w}}, f_{\mathsf{z}}) = D\big(f_{\mathsf{w}} \| p_{\mathsf{w}}\big) - \sum_{m=1}^{M} \left(\ln C_{\mathsf{z}} + \frac{(\hat{z}_m - \hat{p}_m)^2}{2\tau_{p_m}} \right), \tag{4.46}$$

wherein only the C_{z} term depends on $\boldsymbol{\theta}$, suggesting the tuning procedure

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{m=1}^{M} \log \int_{z_m} \mathcal{N}(z_m; \hat{p}_m, \tau_{p_m})\, p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|z_m;\boldsymbol{\theta}). \tag{4.47}$$

Note that (4.47) differs from (4.35) in that the logarithm has moved outside the integral and the distribution over z_m has changed.[9]

Armed with (4.47), we now turn our attention to tuning the probit variance parameter, v. Substituting (4.24) into (4.47), we find

$$v^{k+1} = \underset{v}{\operatorname{argmax}} \sum_{m=1}^{M} \log \int_{z_m} \mathcal{N}(z_m; \hat{p}_m, \tau_{p_m})\, \Phi\left(\frac{y_m z_m}{\sqrt{v}} \right) \tag{4.48}$$

$$= \underset{v}{\operatorname{argmax}} \sum_{m=1}^{M} \log \Phi\left(\frac{y_m \hat{p}_m}{\sqrt{v + \tau_{p_m}}} \right), \tag{4.49}$$

where in going from (4.48) to (4.49), we leveraged the same derivations used to compute the sum-product GAMP updates given in Section 4.4.2. Differentiating

[9] The terms of the secondary summand in (4.35) are dropped since we are assuming that only output channel parameters are being tuned.

(4.49) w.r.t. v and setting equal to zero, we find that

$$0 = \sum_{m=1}^{M} \frac{-c(v^{k+1}) y_m (\hat{z}_m(v^{k+1}) - \hat{p}_m)}{2\tau_{p_m}(v^{k+1} + \tau_{p_m})}, \tag{4.50}$$

with $c(v)$ and $\hat{z}_m(v)$ defined in Table 4.1. Since further optimization of (4.50) is difficult, we resort to Newton's method to iteratively locate v^{k+1}.

A Bethe free entropy-based update of the logistic scale parameter is somewhat more involved; the interested reader can refer to Appendix E.7.

4.6 Numerical Study

In this section we describe several real and synthetic classification problems to which GAMP was applied. Experiments were conducted on a workstation running Red Hat Enterprise Linux (r2.4), with an Intel Core i7-2600 CPU (3.4 GHz, 8MB cache) and 8GB DDR3 RAM.

4.6.1 Text Classification and Adaptive Learning

As a first experiment, we considered a text classification problem drawn from the Reuter's Corpus Volume I (RCV1) dataset [120]. The dataset consists of manually categorized Reuters newswire articles, organized into hierarchical topic codes. We used a version of the dataset tailored to a binary classification problem,[10] in which topic codes `CCAT` and `ECAT` constituted the positive class, and `GCAT` and `MCAT` constituted the negative class; each training example consists of a cosine-normalized log transformation of a TF-IDF (term frequency-inverse document frequency) feature vector. The training dataset consists of 20,242 balanced training examples of

[10] Available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.

$N = 47{,}236$ features, with 677,399 examples reserved for testing, however, following the convention of [101, 121], we swapped training and testing sets in order to test GAMP's computational efficiency on a large training dataset (i.e., $M = 677{,}399$). An interesting feature of the dataset is its sparsity; only 0.16% of the entries in $\boldsymbol{X}$ are non-zero. Given the assumption of a dense, sub-Gaussian $\boldsymbol{X}$ used in deriving the GAMP algorithm, it is not at all obvious that GAMP should even work on a problem of such extreme sparsity. Remarkably, the sparsity does not appear to be problematic, suggesting that the range of problems to which GAMP can be successfully applied may be broader than previously believed.

In Table 4.7 we summarize the performance of several different types of GAMP classifiers, as well as several other popular large-scale linear classifiers. Since linear classifiers are known to perform well in text classification problems [121], and since it is computationally advantageous to exploit the sparsity of $\boldsymbol{X}$, we do not consider kernel-based classifiers.

In the leftmost column of the table, we list the particular pairing of weight vector prior and likelihood used for classification, with the message passing mode (sum-product or max-sum) indicated in parentheses. The next column indicates whether EM learning (with 5 EM iterations), or two-fold cross-validation (over a logarithmically spaced grid of 10 parameter values), was used to set the model parameter(s). The test set accuracy is reported in the next column, followed by runtime, for which we report two numbers. The first number is the total runtime needed to train the classifier, including the EM learning or cross-validation parameter selection process. The second number represents the time needed to train the classifier using the optimal parameter values, as determined via EM or cross-validation. We remark that this latter number is the one most often reported in the literature, but emphasize that it paints an incomplete picture of the true computational cost of training a classifier.

Classifier	EM/x-val	Accuracy (%)	Runtime (s)	Density (%)
Bernoulli-Gaussian + Probit (SP)	EM	97.6	**317** / **57**	11.1
Bernoulli-Gaussian + Hinge (SP)	EM	**97.7**	468 / 93	**8.0**
Laplacian + Logistic (MS)	EM[a]	97.6	684 / 123	9.8
Laplacian + Logistic (MS)	x-val	97.6	3068 / 278	19.6
CDN [101]	x-val	**97.7**	1298 / 112	10.9
TRON [122]	x-val	**97.7**	1682 / 133	10.8
TFOCS (Laplacian + Logistic) [123]	x-val	97.6	1086 / 94	19.2

[a]An approximate EM scheme is used when running GAMP in max-sum mode.

Table 4.7: A comparison of different classifiers (SP: sum-product; MS: max-sum), their parameter tuning approach, test set accuracy, total/optimal runtime, and final model density on the RCV1 binary dataset (w/ training/testing sets flipped).

The final column reports the model density (i.e., the fraction of features selected by the classifier) of the estimated weight vector.[11]

The RCV1 dataset is popular for testing large-scale linear classifiers (see, e.g., [101,121]), and we note that our EM and cross-validation procedures yield accuracies that are competitive with those of other state-of-the-art large-scale linear classifiers, e.g., CDN. We also caution that runtime comparisons between the GAMP classifiers, CDN, and TRON are not apples-to-apples; CDN and TRON are implemented in C++, while GAMP is implemented in MATLAB. Furthermore, while all algorithms used a stopping tolerance of $\varepsilon = 1\times10^{-3}$, their stopping conditions are all slightly different.

4.6.2 Robust Classification

In Section 4.4.4, we proposed an approach by which GAMP can be made robust to labels that are corrupted or otherwise highly atypical under a given activation model

[11]In sum-product mode (which corresponds to MMSE estimation) the estimated weight vector will, in general, have many small, but non-zero, entries. In order to identify the important/discriminative features, we calculate posteriors of the form $\hat{\pi}_n \triangleq p(w_n \neq 0|\boldsymbol{y})$, and include only those features for which $\hat{\pi}_n > 1/2$ in our final model density estimate.

$p^*_{\mathsf{y}|\mathsf{z}}$. We now evaluate the performance of this robustification method. To do so, we first generated examples[12] $(y_m, \boldsymbol{x}_m)$ with balanced classes such that the Bayes-optimal classification boundary is a hyperplane with a desired Bayes error rate of ε_{B}. Then, we flipped a fraction γ of the training labels (but not the test labels), trained several different varieties of GAMP classifiers, and measured their classification accuracy on the test data.

The first classifier we considered paired a genie-aided "standard logistic" activation function, (4.23), with an i.i.d. zero-mean, unit-variance Gaussian weight vector prior. Note that under a class-conditional Gaussian generative distribution with balanced classes, the corresponding activation function is logistic with scale parameter $\alpha = 2M\mu$ [110]. Therefore, the genie-aided logistic classifier was provided the true value of μ, which was used to specify the logistic scale α. The second classifier we considered paired a genie-aided robust logistic activation function, which possessed perfect knowledge of both μ and the mislabeling probability γ, with the aforementioned Gaussian weight vector prior. To understand how performance is impacted by the parameter tuning scheme of Section 4.5, we also trained EM variants of the preceding classifiers. The EM-enabled standard logistic classifier was provided a fixed logistic scale of $\alpha = 100$, and was allowed to tune the variance of the weight vector prior. The EM-enabled robust logistic classifier was similarly configured, and in addition was given an initial mislabeling probability of $\gamma^0 = 0.01$, which was updated according to (4.40).

In Fig. 4.3, we plot the test error rate for each of the four GAMP classifiers as a function of the mislabeling probability γ. For this experiment, μ was set so as to yield

[12] Data was generated according to a class-conditional Gaussian distribution with N discriminatory features. Specifically, given the label $y \in \{-1, 1\}$ a feature vector $\boldsymbol{x}$ was generated as follows: entries of $\boldsymbol{x}$ were drawn i.i.d $\mathcal{N}(y\mu, M^{-1})$ for some $\mu > 0$. Under this model, with balanced classes, the Bayes error rate can be shown to be $\varepsilon_{\mathrm{B}} = \Phi(-\sqrt{NM}\mu)$. The parameter μ can then be chosen to achieve a desired ε_{B}.

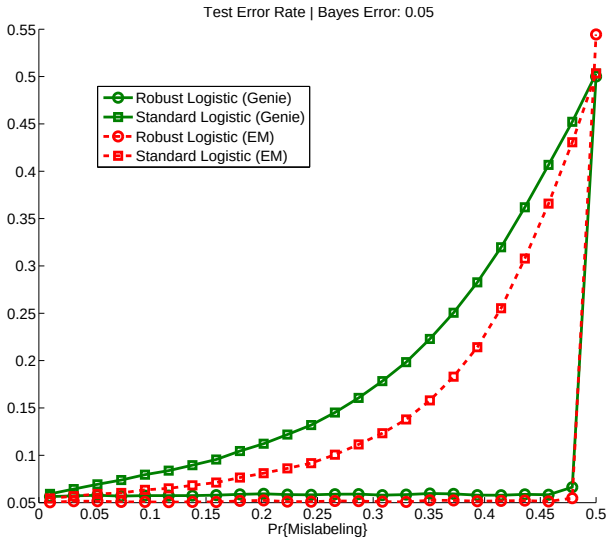


Figure 4.3: Test error rate of genie-aided (solid curves) and EM-tuned (dashed curves) instances of standard logistic ($\square$) and robust logistic ($\circ$) classifiers, as a function of mislabeling probability γ, with $M = 8192$, $N = 512$, and Bayes error rate $\varepsilon_{\mathrm{B}} = 0.05$.

a Bayes error rate of $\varepsilon_{\mathrm{B}} = 0.05$. $M = 8192$ training examples of $N = 512$ training features were generated independently, with the test set error rate evaluated based on 1024 unseen (and uncorrupted) examples. Examining the figure, we can see that EM parameter tuning is beneficial for both the standard and robust logistic classifiers, although the benefit is more pronounced for the standard classifier. Remarkably, both the genie-aided and EM-tuned robust logistic classifiers are able to cope with an extreme amount of mislabeling while still achieving the Bayes error rate, thanks in part to the abundance of training data.

4.6.3 Multi-Voxel Pattern Analysis

Multi-voxel pattern analysis (MVPA) has become an important tool for analyzing functional MRI (fMRI) data [88, 124, 125]. Cognitive neuroscientists, who study how the human brain functions at a physical level, employ MVPA not to infer a subject's

cognitive state but to gather information about how the brain itself distinguishes between cognitive states. In particular, by identifying *which* brain regions are most important in discriminating between cognitive states, they hope to learn the underlying processes by which the brain operates. In this sense, the goal of MVPA is feature selection, not classification.

To investigate the performance of GAMP for MVPA, we conducted an experiment using the well-known Haxby dataset [88]. The Haxby dataset consists of fMRI data collected from 6 subjects with 12 "runs" per subject. In each run, the subject passively viewed 9 greyscale images from each of 8 object categories (i.e., faces, houses, cats, bottles, scissors, shoes, chairs, and nonsense patterns), during which full-brain fMRI data was recorded over $N = 31\,398$ voxels.

In our experiment, we designed classifiers that predict binary object category (e.g., cat vs. scissors) from M examples of N-voxel fMRI data collected from a single subject. For comparison, we tried three algorithms: i) ℓ_1-penalized logistic regression (L1-LR) as implemented using cross-validation-tuned TFOCS [123], ii) L1-LR as implemented using EM-tuned max-sum GAMP, and iii) sum-product GAMP under a Bernoulli-Laplace prior and logistic activation function (BL-LR).

Algorithm performance (i.e., error-rate, sparsity, consistency) was assessed using 12-fold leave-one-out cross-validation. In other words, for each algorithm, 12 separate classifiers were trained, each for a different combination of 1 testing fold (used to evaluate error-rate) and 11 training folds. The reported performance then represents an average over the 12 classifiers. Each fold comprised one of the runs described above, and thus contained 18 examples (i.e., 9 images from each of the 2 object categories constituting the pair), yielding a total of $M = 11 \times 18 = 198$ training examples. Since $N = 31\,398$, the underlying problem is firmly in the $N \gg M$ regime.

To tune each TFOCS classifier (i.e., select its ℓ_1 regularization weight λ), we used

Comparison	Error Rate (%)			Sparsity (%)		
	TFOCS	L1-LR	BL-LR	TFOCS	L1-LR	BL-LR
Cat vs. Scissors	**9.7**	11.1	14.8	0.1	0.07	**0.03**
Cat vs. Shoe	**6.1**	**6.1**	20.8	0.14	0.07	**0.03**
Cat vs. House	0.4	**0.0**	0.9	0.04	**0.02**	0.03
Bottle vs. Shoe	**29.6**	30.5	33.3	0.2	0.1	**0.04**
Bottle vs. Chair	13.9	13.9	**13.4**	0.1	0.07	**0.03**
Face vs. Chair	**0.9**	**0.9**	5.1	0.09	0.045	**0.03**

Comparison	Consistency (%)			Runtime (s)		
	TFOCS	L1-LR	BL-LR	TFOCS	L1-LR	BL-LR
Cat vs. Scissors	38	**43**	23	1318	**137**	158
Cat vs. Shoe	34	**47**	15	1347	191	**154**
Cat vs. House	53	**87**	52	1364	144	**125**
Bottle vs. Shoe	23	**31**	7	1417	**166**	186
Bottle vs. Chair	30	**45**	37	1355	**150**	171
Face vs. Chair	43	**67**	25	1362	**125**	164

Table 4.8: Performance of L1-LR TFOCS ("TFOCS"), L1-LR EM-GAMP ("L1-LR"), and BL-LR EM-GAMP ("BL-LR") classifiers on various Haxby pairwise comparisons.

a second level of leave-one-out cross-validation. For this, we first chose a fixed $G = 10$-element grid of logarithmically spaced λ hypotheses. Then, for each hypothesis, we designed 11 TFOCS classifiers, each of which used 10 of the 11 available folds for training and the remaining fold for error-rate evaluation. Finally, we chose the λ hypothesis that minimized the error-rate averaged over these 11 TFOCS classifiers. For EM-tuned GAMP, there was no need to perform the second level of cross-validation: we simply applied the EM tuning strategy described in Section 4.5 to the 11-fold training data.

Table 4.8 reports the results of the above-described experiment for six pairwise comparisons. There, sparsity refers to the average percentage of non-zero[13] elements

[13]The weight vectors learned by sum-product GAMP contained many entries that were very small but not exactly zero-valued. Thus, the sparsity reported in Table 4.8 is the percentage of weight-vector entries that contained 99% of the weight-vector energy.

in the learned weight vectors. Consistency refers to the average Jaccard index between weight-vector supports, i.e.,

$$\text{consistency} := \frac{1}{12}\sum_{i=1}^{12}\frac{1}{11}\sum_{j\neq i}\frac{|\mathcal{S}_i \cap \mathcal{S}_j|}{|\mathcal{S}_i \cup \mathcal{S}_j|} \tag{4.51}$$

where $\mathcal{S}_i$ denotes the support of the weight vector learned when holding out the i^{th} fold. Runtime refers to the total time used to complete the 12-fold cross-validation procedure.

Ideally, we would like an algorithm that computes weight vectors with low cross-validated error rate, that are very sparse, that are consistent across folds, and that are computed very quickly. Although Table 4.8 reveals no clear winner, it does reveal some interesting trends. Comparing the results for TFOCS and EM-GAMP (which share the L1-LR objective but differ in minimization strategy and tuning), we see similar error rates. However, EM-GAMP produced classifiers that were uniformly more sparse, more consistent, and did so with runtimes that were almost an order-of-magnitude faster. We attribute the faster runtimes to the tuning strategy, since cross-validation-tuning required the design of 11 TFOCS classifiers for every EM-GAMP classifier. In comparing BL-LR to the other two algorithms, we see that its error rates are not as good in most cases, but that the resulting classifiers were usually much sparser. The runtime of BL-LR is similar to that of L1-LR GAMP, which is not surprising since both use the same EM-based tuning scheme.

4.7 Conclusion

In this chapter, we presented the first comprehensive study of the *generalized approximate message passing* (GAMP) algorithm [27] in the context of linear binary

classification. We established that a number of popular discriminative models, including logistic and probit regression, and support vector machines, can be implemented in an efficient manner using the GAMP algorithmic framework, and that GAMP's state evolution formalism can be used in certain instances to predict the misclassification rate of these models. In addition, we demonstrated that a number of sparsity-promoting weight vector priors can be paired with these activation functions to encourage feature selection. Importantly, GAMP's message passing framework enables us to learn the hyperparameters that govern our probabilistic models adaptively from the data using expectation-maximization (EM), a trait which can be advantageous when cross-validation proves infeasible. The flexibility imparted by the GAMP framework allowed us to consider several modifications to the basic discriminative models, such as robust classification, which can be effectively implemented using existing non-robust modules. Moreover, by embedding GAMP within a larger probabilistic graphical model, it is possible to consider a wide variety of structured priors on the weight vector, e.g., priors that encourage spatial clustering of important features.

In a numerical study, we confirmed the efficacy of our approach on both real and synthetic classification problems. For example, we found that the proposed EM parameter tuning can be both computationally efficient and accurate in the applications of text classification and multi-voxel pattern analysis. We also observed on synthetic data that the robust classification extension can substantially outperform a non-robust counterpart.

BIBLIOGRAPHY

[1] International Data Corp., "Extracting value from chaos." http://www.emc.com/about/news/press/2011/20110628-01.htm, Jun. 2011.

[2] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. Inform. Theory*, vol. 52, pp. 489 – 509, Feb. 2006.

[3] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, pp. 1289–1306, Apr. 2006.

[4] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, pp. 21–30, Mar. 2008.

[5] M. A. Davenport, M. F. Duarte, Y. C. Eldar, and G. Kutyniok, *Compressed Sensing: Theory and Applications*, ch. Introduction to Compressed Sensing. Cambridge Univ. Press, 2012.

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc., B*, vol. 58, no. 1, pp. 267–288, 1996.

[7] B. D. Rao and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," in *IEEE Digital Signal Process. Workshop*, (Bryce Canyon, UT), pp. 1–4, June 1998.

[8] W. Lu and N. Vaswani, "Modified compressive sensing for real-time dynamic MR imaging," in *IEEE Int'l Conf. Image Processing (ICIP)*, pp. 3045 –3048, Nov. 2009.

[9] W. Li and J. C. Preisig, "Estimation of rapidly time-varying sparse channels," *IEEE J. Oceanic Engr.*, vol. 32, pp. 927–939, Oct. 2007.

[10] S. Mallat, *A Wavelet Tour of Signal Processing*. San Diego, CA: Academic Press, 2008.

[11] M. F. Duarte and Y. C. Eldar, "Structured compressed sensing: From theory to applications," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4053–4085, 2011.

[12] S. F. Cotter, B. D. Rao, K. Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *IEEE Trans. Signal Process.*, vol. 53, pp. 2477–2488, Jul. 2005.

[13] D. P. Wipf and B. D. Rao, "An empirical Bayesian strategy for solving the simultaneous sparse approximation problem," *IEEE Trans. Signal Process.*, vol. 55, pp. 3704–3716, Jul. 2007.

[14] Z. Zhang and B. D. Rao, "Sparse signal recovery with temporally correlated source vectors using Sparse Bayesian Learning," *IEEE J. Selected Topics Signal Process.*, vol. 5, pp. 912–926, Sept. 2011.

[15] P. Schniter, "A message-passing receiver for BICM-OFDM over unknown clustered-sparse channels," *IEEE J. Select. Topics in Signal Process.*, vol. 5, pp. 1462–1474, Dec. 2011.

[16] S. Som and P. Schniter, "Compressive imaging using approximate message passing and a Markov-tree prior," *IEEE Trans. Signal Process.*, vol. 60, pp. 3439–3448, Jul. 2012.

[17] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[18] B. Shahrasbi, A. Talari, and N. Rahnavard, "TC-CSBP: Compressive sensing for time-correlated data based on belief propagation," in *Conf. on Inform. Sci. and Syst. (CISS)*, (Baltimore, MD), pp. 1–6, Mar. 2011.

[19] J. P. Vila and P. Schniter, "Expectation-Maximization Gaussian-mixture approximate message passing," in *Proc. Conf. on Information Sciences and Systems*, (Princeton, NJ), Mar. 2012.

[20] D. Sejdinović, C. Andrieu, and R. Piechocki, "Bayesian sequential compressed sensing in sparse dynamical systems," in *48th Allerton Conf. Comm., Control, & Comp.*, (Urbana, IL), pp. 1730–1736, Nov. 2010.

[21] S. Shedthikere and A. Chockalingam, "Bayesian framework and message passing for joint support and signal recovery of approximately sparse signals," in *IEEE Int'l Conf. Acoust., Speech & Signal Process. (ICASSP)*, (Prague, Czech Republic), pp. 4032–4035, May 2011.

[22] B. Mailhé, R. Gribonval, P. Vandergheynst, and F. Bimbot, "Fast orthogonal sparse approximation algorithms over local dictionaries," *Signal Processing*, vol. 91, no. 12, pp. 2822 – 2835, 2011.

[23] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Trans. Signal Process.*, vol. 56, pp. 2346 –2356, June 2008.

[24] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.

[25] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," in *Proceedings of the National Academy of Sciences*, vol. 106, pp. 18914–18919, Nov. 2009.

[26] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. motivation and construction," in *Proc. of Information Theory Workshop*, Jan. 2010.

[27] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Int'l Symp. Inform. Theory*, (St. Petersburg, Russia), pp. 2168–2172, Aug. 2011. (See also arXiv: 1010.5141).

[28] B. J. Frey and D. J. C. MacKay, "A revolution: Belief propagation in graphs with cycles," *Neural Information Processing Systems (NIPS)*, vol. 10, pp. 479–485, 1998.

[29] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Trans. Inform. Theory*, vol. 57, pp. 764–785, Feb. 2011.

[30] 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.

[31] J. Ziniel and P. Schniter, "Efficient message passing-based inference in the multiple measurement vector problem," in *Proc. 45th Asilomar Conf. Sig., Sys., & Comput. (SS&C)*, (Pacific Grove, CA), Nov. 2011.

[32] J. Ziniel and P. Schniter, "Efficient high-dimensional inference in the multiple measurement vector problem," *IEEE Trans. Signal Process.*, vol. 61, pp. 340–354, Jan. 2013.

[33] G. Tzagkarakis, D. Milioris, and P. Tsakalides, "Multiple-measurement Bayesian compressed sensing using GSM priors for DOA estimation," in *IEEE Int'l Conf. Acoust., Speech & Signal Process. (ICASSP)*, (Dallas, TX), pp. 2610–2613, Mar. 2010.

[34] D. Liang, L. Ying, and F. Liang, "Parallel MRI acceleration using M-FOCUSS," in *Int'l Conf. Bioinformatics & Biomed. Eng. (ICBBE)*, (Beijing, China), pp. 1–4, June 2009.

[35] Y. Eldar and H. Rauhut, "Average case analysis of multichannel sparse recovery using convex relaxation," *IEEE Trans. Inform. Theory*, vol. 56, pp. 505–519, Jan. 2010.

[36] J. Ziniel, L. C. Potter, and P. Schniter, "Tracking and smoothing of time-varying sparse signals via approximate belief propagation," in *Asilomar Conf. on Signals, Systems and Computers 2010*, (Pacific Grove, CA), Nov. 2010.

[37] J. Ziniel and P. Schniter, "Dynamic compressive sensing of time-varying signals via approximate message passing," *IEEE Trans. Signal Process.*, vol. 61, pp. 5270–5284, Nov. 2013.

[38] Y. Eldar and M. Mishali, "Robust recovery of signals from a structured union of subspaces," *IEEE Trans. Inform. Theory*, vol. 55, pp. 5302–5316, Nov. 2009.

[39] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation. Part II: Convex relaxation," *Signal Processing*, vol. 86, pp. 589–602, Apr. 2006.

[40] M. M. Hyder and K. Mahata, "A robust algorithm for joint sparse recovery," *IEEE Signal Process. Lett.*, vol. 16, pp. 1091–1094, Dec. 2009.

[41] Z. Zhang and B. D. Rao, "Iterative reweighted algorithms for sparse signal recovery with temporally correlated source vectors," in *IEEE Int'l Conf. Acoust., Speech & Signal Process. (ICASSP)*, (Prague, Czech Republic), pp. 3932–3935, May 2011.

[42] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, "Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit," *Signal Processing*, vol. 86, pp. 572–588, Apr. 2006.

[43] K. Lee, Y. Bresler, and M. Junge, "Subspace methods for joint sparse recovery," *IEEE Trans. Inform. Theory*, vol. 58, no. 6, pp. 3613–3641, 2012.

[44] J. Kim, W. Chang, B. Jung, D. Baron, and J. C. Ye, "Belief propagation for joint sparse recovery." arXiv:1102.3289v1 [cs.IT], Feb. 2011.

[45] N. Vaswani, "Kalman filtered compressed sensing," in *IEEE Int'l Conf. on Image Processing (ICIP) 2008*, pp. 893 –896, 12-15 2008.

[46] D. Angelosante, G. B. Giannakis, and E. Grossi, "Compressed sensing of time-varying signals," in *Int'l Conf. on Digital Signal Processing 2009*, pp. 1–8, July 2009.

[47] D. Angelosante, S. Roumeliotis, and G. Giannakis, "Lasso-Kalman smoother for tracking sparse signals," in *Asilomar Conf. on Signals, Systems and Computers 2009*, (Pacific Grove, CA), pp. 181 – 185, Nov. 2009.

[48] N. Vaswani and W. Lu, "Modified-CS: Modifying compressive sensing for problems with partially known support," *IEEE Trans. Signal Process.*, vol. 58, pp. 4595–4607, Sept. 2010.

[49] E. van den Berg and M. P. Friedlander, "Theoretical and empirical results for recovery from multiple measurements," *IEEE Trans. Inform. Theory*, vol. 56, pp. 2516–2527, May 2010.

[50] D. L. Donoho, I. Johnstone, and A. Montanari, "Accurate prediction of phase transitions in compressed sensing via a connection to minimax denoising," *IEEE Trans. Inform. Theory*, vol. 59, Jun. 2013.

[51] W. U. Bajwa, J. Haupt, A. M. Sayeed, and R. Nowak, "Compressed channel sensing: A new approach to estimating sparse multipath channels," *Proc. IEEE*, vol. 98, pp. 1058–1076, Jun. 2010.

[52] M. A. Ohliger and D. K. Sodickson, "An introduction to coil array design for parallel MRI," *NMR in Biomed.*, vol. 19, pp. 300–315, May 2006.

[53] P. Schniter, "Turbo reconstruction of structured sparse signals," in *Conf. on Information Sciences and Systems (CISS)*, pp. 1 –6, Mar. 2010.

[54] R. L. Eubank, *A Kalman Filter Primer*. Boca Raton, FL: Chapman & Hall/CRC, 2006.

[55] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, pp. 498–519, Feb. 2001.

[56] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufman, 1988.

[57] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *Int'l. J. Comp. Vision*, vol. 40, pp. 25–47, Oct. 2000.

[58] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. New York: Cambridge University Press, 2003.

[59] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE Trans. Signal Process.*, vol. 58, pp. 269 –280, Jan. 2010.

[60] S. Som and P. Schniter, "Approximate message passing for recovery of sparse signals with Markov-random-field support structure," in *Int'l Conf. Mach. Learn.*, (Bellevue, Wash.), Jul. 2011.

[61] J. Ziniel, S. Rangan, and P. Schniter, "A generalized framework for learning and recovery of structured sparse signals," in *Proc. Stat. Signal Process. Wkshp*, (Ann Arbor, MI), Aug. 2012.

[62] O. Zoeter and T. Heskes, "Change point problems in linear dynamical systems," *J. Mach. Learn. Res.*, vol. 6, pp. 1999–2026, Dec. 2005.

[63] D. L. Alspach and H. W. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Trans. Auto. Control*, vol. 17, pp. 439–448, Aug. 1972.

[64] D. Barber and A. T. Cemgil, "Graphical models for time series," *IEEE Signal Process. Mag.*, vol. 27, pp. 18–28, Nov. 2010.

[65] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc. B*, vol. 39, pp. 1–38, 1977.

[66] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Process. Mag.*, vol. 13, pp. 47–60, Nov. 1996.

[67] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer-Verlag, 2006.

[68] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, pp. 2173–2200, Oct. 2001.

[69] M. Salman Asif, D. Reddy, P. Boufounos, and A. Veeraraghavan, "Streaming compressive sensing for high-speed periodic videos," in *Int'l Conf. on Image Processing (ICIP) 2010*, (Hong Kong), Sept. 2010.

[70] N. Vaswani, "LS-CS-residual (LS-CS): Compressive sensing on least squares residual," *IEEE Trans. Signal Process.*, vol. 58, no. 8, pp. 4108–4120, 2010.

[71] S. Das and N. Vaswani, "Particle filtered modified compressive sensing (PaFiMoCS) for tracking signal sequences," in *Asilomar Conf. on Signals, Systems and Computers 2010*, pp. 354–358, Nov. 2010.

[72] W. Lu and N. Vaswani, "Regularized modified bpdn for noisy sparse reconstruction with partial erroneous support and signal value knowledge," *Signal Processing, IEEE Transactions on*, vol. 60, no. 1, pp. 182–196, 2012.

[73] W. Dai, D. Sejdinović, and O. Milenkovic, "Gaussian dynamic compressive sensing," in *Int'l Conf. on Sampling Theory and Appl. (SampTA)*, (Singapore), May 2011.

[74] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer, 2nd ed., 2004.

[75] S. C. Tatikonda and M. I. Jordan, *Loopy belief propagation and Gibbs measures*, pp. 493–500. Proc. 18th Conf. Uncertainty in Artificial Intelligence (UAI), San Mateo, CA: Morgan Kaufmann, 2002.

[76] T. Heskes, "On the uniqueness of belief propagation fixed points," *Neural Comput.*, vol. 16, no. 11, pp. 2379–2413, 2004.

[77] A. T. Ihler, J. W. Fisher III, and A. S. Willsky, "Loopy belief propagation: Convergence and effects of message errors," *J. Mach. Learn. Res.*, vol. 6, pp. 905–936, 2005.

[78] R. J. McEliece, D. J. C. MacKay, and J. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *IEEE J. Select. Areas Comm.*, vol. 16, pp. 140–152, Feb. 1998.

[79] G. Elidan, I. McGraw, and D. Koller, "Residual belief propagation: Informed scheduling for asynchronous message passing," in *Proc. 22nd Conf. Uncertainty Artificial Intelligence (UAI)*, 2006.

[80] D. L. Donoho and J. Tanner, "Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing," *Phil. Trans. R. Soc. A*, vol. 367, pp. 4273–4293, 2009.

[81] H. A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang, "The factor graph approach to model-based signal processing," *Proc. of the IEEE*, vol. 95, no. 6, pp. 1295–1322, 2007.

[82] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Scientific Comp.*, vol. 20, no. 1, pp. 33–61, 1998.

[83] E. van den Berg and M. Friedlander, "Probing the Pareto frontier for basis pursuit solutions," *SIAM J. Scientific Comp.*, vol. 31, pp. 890–912, Nov. 2008.

[84] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Allerton Conf. on Comm., Control, and Comp.*, pp. 704 –711, Oct. 2010.

[85] Y. Chi, Y. Eldar, and R. Calderbank, "PETRELS: Subspace estimation and tracking from partial observations," in *Int'l Conf. Acoustics, Speech, & Signal Process. (ICASSP)*, (Kyoto, Japan), Mar. 2012.

[86] J. Ziniel, P. Sederberg, and P. Schniter, "Binary classification and feature selection via generalized approximate message passing." In submission: *IEEE Trans. Inform. Theory*, arXiv:1401.0872 [cs.IT], 2014.

[87] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *J. Mach. Learn. Res.*, vol. 3, pp. 1289–1305, 2003.

[88] J. V. Haxby, M. I. Gobbini, M. L. Furey, A. Ishai, J. L. Schouten, and P. Pietrini, "Distributed and overlapping representations of faces and objects in ventral temporal cortex," *Science*, vol. 293, pp. 2425–2430, Sept. 2001.

[89] S. Ryali, K. Supekar, D. A. Abrams, and V. Menon, "Spase logistic regression for whole-brain classification of fMRI data," *NeuroImage*, vol. 51, pp. 752–764, 2010.

[90] A. M. Chan, E. Halgren, K. Marinkovic, and S. S. Cash, "Decoding word and category-specific spatiotamporal representations from MEG and EEG," *NeuroImage*, vol. 54, pp. 3028–3039, 2011.

[91] A. Gustafsson, A. Hermann, and F. Huber, *Conjoint Measurement: Methods and Applications*. Berlin: Springer-Verlag, 2007.

[92] E. P. Xing, M. I. Jordan, and R. M. Karp, "Feature selection for high-dimensional genomic microarray data," in *Int'l Wkshp. Mach. Learn.*, pp. 601–608, 2001.

[93] P. T. Boufounos and R. G. Baraniuk, "1-bit compressive sensing," in *Proc. Conf. Inform. Science & Sys.*, (Princeton, NJ), Mar. 2008.

[94] Y. Plan and R. Vershynin, "Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach," *IEEE Trans. Inform. Theory*, vol. 59, no. 1, pp. 482–494, 2013.

[95] D. Koller and M. Sahami, "Toward optimal feature selection," in *Proc. 13th Int'l Conf. Machine Learning (ICML)* (L. Saitta, ed.), (Bari, Italy), pp. 284–292, 1996.

[96] R. Kohavi and G. John, "Wrapper for feature subset selection," *Artificial Intell.*, vol. 97, pp. 273–324, 1997.

[97] M. Figueiredo, "Adaptive sparseness using Jeffreys' prior," in *Proc. 14th Conf. Advances Neural Inform. Process. Sys.*, pp. 697–704, MIT Press, Cambridge, MA, 2001.

[98] M. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 25, no. 9, pp. 1150–1159, 2003.

[99] A. Kabán, "On Bayesian classification with Laplace priors," *Pattern Recognition Lett.*, vol. 28, no. 10, pp. 1271–1282, 2007.

[100] H. Chen, P. Tino, and X. Yao, "Probabilistic classification vector machines," *IEEE Trans. Neural Net.*, vol. 20, no. 6, pp. 901–914, 2009.

[101] G.-X. Yuan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin, "A comparison of optimization methods and software for large-scale L1-regularized linear classification," *J. Mach. Learn. Res.*, vol. 11, pp. 3183–3234, 2010.

[102] A. Gupta, R. Nowak, and B. Recht, "Sample complexity for 1-bit compressed sensing and sparse classification," in *Proc. Int'l Symp. Inform Theory (ISIT)*, (Austin, TX), 2010.

[103] J. N. Laska, Z. Wen, W. Yin, and R. G. Baraniuk, "Trust, but verify: Fast and accurate signal recovery from 1-bit compressive measurements," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5289–5301, 2011.

[104] U. S. Kamilov, A. Bourquard, A. Amini, and M. Unser, "One-bit measurements with adaptive thresholds," *IEEE Signal Process. Lett.*, vol. 19, pp. 607–610, 2012.

[105] U. S. Kamilov, V. K. Goyal, and S. Rangan, "Message-passing de-quantization with applications to compressed sensing," vol. 60, pp. 6270–6281, Dec. 2012.

[106] J. P. Vila and P. Schniter, "Expectation-Maximization Gaussian-mixture approximate message passing," *IEEE Trans. Signal Process.*, vol. 61, pp. 4658–4672, Oct. 2013.

[107] S. Rangan, P. Schniter, E. Riegler, A. Fletcher, and V. Cevher, "Fixed points of generalized approximate message passing with arbitrary matrices," in *Int'l Symp. Inform. Theory (ISIT)*, pp. 664–668, IEEE, 2013.

[108] S. Rangan, P. Schniter, and A. Fletcher, "On the convergence of generalized approximate message passing with arbitrary matrices," (Honolulu, Hawaii), July 2014. (Full version at *arXiv:1402.3210*).

[109] H. A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, pp. 28–41, Jan. 2004.

[110] M. I. Jordan, "Why the logistic function? A tutorial discussion on probabilities and neural networks," 1995.

[111] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[112] B. E. Boser, I. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Wkshp Computational Learn. Theory*, pp. 144–152, ACM Press, 1992.

[113] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY: Springer-Verlag, 1995.

[114] M. Opper and O. Winther, *Gaussian Processes and SVM: Mean Field Results and Leave-One-Out Estimator*, ch. 17, pp. 311–326. MIT Press, 2000.

[115] A. K. Nigam, K.and McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, pp. 103–134, 2000.

[116] J. Vila and P. Schniter, "An empirical-Bayes approach to recovering linearly constrained non-negative sparse signals," *IEEE Trans. Signal Process.*, vol. 62, pp. 4689–4703, Sep. 2014.

[117] U. S. Kamilov, S. Rangan, A. K. Fletcher, and M. Unser, "Approximate message passing with consistent parameter estimation and applications to sparse learning," in *Proc. Neural Inform. Process. Syst. Conf.*, (Lake Tahoe, NV), Dec. 2012. (Full version at *arXiv:1207.3859*).

[118] F. Krzakala, M. Mézard, and L. Zdeborová, "Phase diagram and approximate message passing for blind calibration and dictionary learning," in *Int'l Symp. Inform. Theory (ISIT)*, pp. 659–663, IEEE, 2013.

[119] F. Krzakala, M. Mézard, F. Sausset, Y. Sun, and L. Zdeborová, "Probabilistic reconstruction in compressed sensing: algorithms, phase diagrams, and threshold achieving matrices," *J. Stat. Mechanics*, vol. 2012, no. 08, p. P08009, 2012.

[120] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, 2004.

[121] C. Lin, R. C. Weng, and S. S. Keerthi, "Trust region Newton methods for large-scale logistic regression," in *Proc. 24th Int'l Conf. Mach. Learn.*, (Corvallis, OR), pp. 561–568, 2007.

[122] C. J. Lin and J. J. Moré, "Newton's method for large-scale bound constrained problems," *SIAM J. Optimization*, vol. 9, pp. 1100–1127, 1999.

[123] S. R. Becker, E. J. Candès, and M. C. Grant, "Templates for convex cone problems with applications to sparse signal recovery," *Math. Prog. Comp.*, vol. 3, no. 3, pp. 165–218, 2011.

[124] 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, pp. 424–430, Sep. 2006.

[125] F. Pereira, T. Mitchell, and M. Botvinick, "Machine learning classifiers and fMRI: A tutorial overview," *NeuroImage*, vol. 45, pp. S199–S209, Mar. 2009.

[126] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing." arXiv:1010.5141v1 [cs.IT], 2010.

[127] P. Schniter and S. Rangan, "Compressive phase retrieval via generalized approximate message passing," in *Proc. Allerton Conf. on Communication, Control, & Computing*, (Monticello, IL), Oct. 2012.

[128] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," *International Comp. Sci. Inst.*, vol. 4, no. 510, p. 126, 1998.

[129] D. R. Barr and E. T. Sherrill, "Mean and variance of truncated normal distributions," *American Statistician*, vol. 53, Nov. 1999.

Appendix A

THE BASICS OF BELIEF PROPAGATION AND (G)AMP

In this appendix, we provide a brief primer on belief propagation, the approximate message passing (AMP) algorithmic framework proposed by Donoho, Maleki, and Montanari [25,26], and the generalized AMP (GAMP) framework developed by Rangan [27, 126].[1] To begin with, we consider the task of estimating a signal vector $\boldsymbol{x} \in \mathbb{C}^N$ from linearly compressed and AWGN-corrupted measurements:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} \in \mathbb{C}^M. \tag{A.1}$$

AMP can be derived from the perspective of loopy belief propagation (LBP) [28, 55], a Bayesian inference strategy that is based on a factorization of the signal posterior pdf, $p(\boldsymbol{x}|\boldsymbol{y})$, into a product of simpler pdfs that, together, reveal the probabilistic structure in the problem. Concretely, if the signal coefficients, $\boldsymbol{x}$, and noise samples, $\boldsymbol{e}$, in (A.1) are jointly independent such that $p_{\boldsymbol{x}}(\boldsymbol{x}) = \prod_{n=1}^{N} p_x(x_n)$ and $p_{\boldsymbol{y}|\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{z}) = \prod_{m=1}^{M} p_{y|z}(y_m|z_m) = \prod_{m=1}^{M} \mathcal{CN}(y_m; z_m, \sigma_e^2)$, where $z_m \triangleq \boldsymbol{a}_m^{\mathsf{T}}\boldsymbol{x}$, then the posterior pdf factors as

$$p(\boldsymbol{x}|\boldsymbol{y}) \propto \prod_{m=1}^{M} \mathcal{CN}(y_m; \boldsymbol{a}_m^{\mathsf{T}}\boldsymbol{x}, \sigma_e^2) \prod_{n=1}^{N} p_x(x_n), \tag{A.2}$$

[1] Portions of this primer are courtesy of material first published in [127].

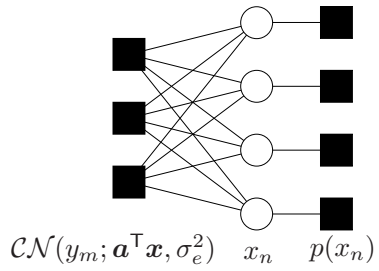


Figure A.1: The factor graph representation of the decomposition of (A.2).

yielding the factor graph in Fig. A.1.

In belief propagation [56], messages representing beliefs about the unknown variables are exchanged amongst the nodes of the factor graph until convergence to a stable fixed point occurs. The set of beliefs passed into a given variable node are then used to infer statistical properties of the associated random variable, e.g., the posterior mode, or a complete posterior distribution. The sum-product algorithm [55] is perhaps the most well-known approach to belief propagation, wherein the messages take the form of probability distributions, and exact posteriors are guaranteed whenever the graph does not have cycles ("loops"). For graphs with cycles, exact inference is known to be NP-hard, and so LBP is not guaranteed to produce correct posteriors. Still, it has shown state-of-the-art performance on a wide array of challenging inference problems, as noted in Section 3.3.2.

The conventional wisdom surrounding LBP says that accurate inference is possible only when the factor graph is locally tree-like, i.e., the girth of any cycle is relatively large. With (A.1), this would require that $\boldsymbol{A}$ is an appropriately constructed sparse matrix, which precludes some of the most interesting CS problems. Surprisingly, in recent work, it was established that LBP-inspired compressive sensing, via AMP, is both feasible [25, 26] for dense $\boldsymbol{A}$ matrices, and provably accurate [29]. In particular,

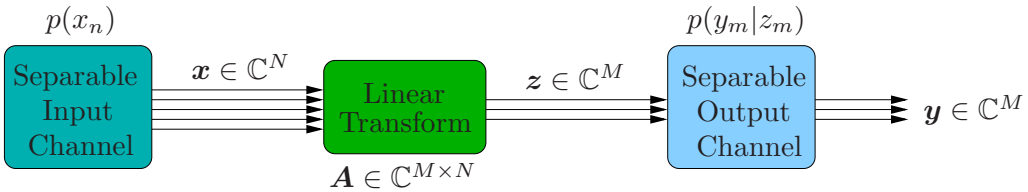


Figure A.2: The GAMP system model.

in the large-system limit (i.e., as $M, N \to \infty$ with M/N fixed) and under i.i.d. sub-Gaussian $\boldsymbol{A}$, the iterations of AMP are governed by a state-evolution whose fixed point—when unique—yields the true posterior means. Interestingly, not only can AMP solve the compressive sensing problem (A.1), but it can do so much faster, and more accurately, than other state-of-the-art methods, whether optimization-based, greedy, or Bayesian. To accomplish this feat, [25, 26] proposed a specific set of approximations that become accurate in the limit of large, dense $\boldsymbol{A}$ matrices, yielding algorithms that give accurate results using only $\approx 2MN$ flops-per-iteration, and relatively few iterations (e.g., tens).

Generalized AMP (GAMP) extends the AMP framework to settings in which the relationship between y_m and z_m is (possibly) nonlinear. The system model for GAMP is illustrated in Fig. A.2. The difference between this system model, and that of AMP, is that GAMP allows for arbitrary separable "output channels," $p_{y|z}(y_m|z_m)$. This is advantageous when considering categorical output variables, $\{y_m\}$, as in the classification problem described in Chapter 4.

The specific implementation of any (G)AMP algorithm will depend on the particular choices of likelihood, $p_{y|z}(y|z)$, and prior, $p_x(x)$, but ultimately amounts to an iterative, scalar soft-thresholding procedure with a carefully chosen adaptive thresholding strategy. Deriving the appropriate thresholding functions for a particular

signal model can be accomplished by computing scalar sum-product, or max-sum, updates of a simple form (see, e.g., Algorithm 1 of Section 4.2 for the generic GAMP algorithm).

Appendix B

TAYLOR SERIES APPROXIMATION OF $\nu^{\mathbf{MOD}}_{F_n^{(t)} \rightarrow \theta_n^{(t)}}$

In this appendix we summarize the procedure used to collapse the binary Gaussian mixture of (2.12) to a single Gaussian. For simplicity, we drop the n and (t) sub- and superscripts.

Let $\theta_r \triangleq \mathfrak{Re}\{\theta\}$, let $\theta_i \triangleq \mathfrak{Im}\{\theta\}$, and let ϕ_r and ϕ_i be defined similarly. Define

$$\begin{aligned}
\tilde{g}(\theta_r, \theta_i) &\triangleq \nu^{\text{mod}}_{f \rightarrow \theta}(\theta_r + j\theta_i), \\
&= (1 - \Omega(\overleftarrow{\pi}))\ \mathcal{CN}(\theta_r + j\theta_i; \tfrac{1}{\varepsilon}\phi, \tfrac{1}{\varepsilon^2}c) + \Omega(\overleftarrow{\pi})\ \mathcal{CN}(\theta_r + j\theta_i; \phi, c) \\
\tilde{f}(\theta_r, \theta_i) &\triangleq -\log \tilde{g}(\theta_r, \theta_i).
\end{aligned}$$

Our objective is to approximate $\tilde{f}(\theta_r, \theta_i)$ using a two-dimensional second-order Taylor series expansion, $\breve{f}(\theta_r, \theta_i)$, about the point ϕ:

$$\begin{aligned}
\breve{f}(\theta_r, \theta_i) &= \tilde{f}(\phi_r, \phi_i) + (\theta_r - \phi_r)\frac{\partial \tilde{f}}{\partial \theta_r} + (\theta_i - \phi_i)\frac{\partial \tilde{f}}{\partial \theta_i} \\
&\quad + \frac{1}{2}\left[(\theta_r - \phi_r)^2 \frac{\partial^2 \tilde{f}}{\partial \theta_r^2} + (\theta_r - \phi_r)(\theta_i - \phi_i)\frac{\partial^2 \tilde{f}}{\partial \theta_r \partial \theta_i} + (\theta_i - \phi_i)^2 \frac{\partial^2 \tilde{f}}{\partial \theta_i^2}\right],
\end{aligned}$$

with all partial derivatives evaluated at ϕ. It can be shown that, for Taylor series expansions about the point ϕ, $\frac{\partial^2 f}{\partial \theta_r \partial \theta_i} = \mathcal{O}(\varepsilon^2)$ and $\left|\frac{\partial^2 f}{\partial \theta_r^2} - \frac{\partial^2 f}{\partial \theta_i^2}\right| = \mathcal{O}(\varepsilon^2)$. Since $\varepsilon \ll 1$, it is reasonable to therefore adopt a further approximation and assume $\frac{\partial^2 \tilde{f}}{\partial \theta_r \partial \theta_i} = 0$ and

$\frac{\partial^2 \tilde{f}}{\partial \theta_r^2} = \frac{\partial^2 \tilde{f}}{\partial \theta_i^2}$. With this approximation, note that

$$\exp(-\breve{f}(\theta_r, \theta_i)) \propto \mathcal{CN}(\theta_r + j\theta_i; \vec{\xi}, \vec{\psi}),$$

with

$$\vec{\psi} \triangleq 2\frac{\partial^2 \tilde{f}}{\partial \theta_r^2}^{-1}, \tag{B.1}$$

$$\vec{\xi} \triangleq \phi_r + j\phi_i - \frac{\vec{\psi}}{2}\left(\frac{\partial \tilde{f}}{\partial \theta_r} + j\frac{\partial \tilde{f}}{\partial \theta_i}\right). \tag{B.2}$$

The pseudocode function, taylor_approx, that computes (B.1), (B.2) given the parameters of $\nu_{f\to\theta}^{\text{mod}}(\cdot)$ is provided in Table 2.3.

Appendix C

DCS-AMP MESSAGE DERIVATIONS

In this appendix, we provide derivations of the various messages needed to implement the DCS-AMP algorithm, as summarized in the pseudocode of Table 3.2. To aid our derivations, in Fig. C.1 we reproduce the message summary figure from Section 3.3.2. Directed edges indicate the direction that messages are moving. In the **(across)** phase, we only illustrate the messages involved in a forward pass for the amplitude variables, and leave out a graphic for the corresponding backward pass, as well as graphics for the support variable **(across)** phase. Note that, to be applicable at frame T, the factor node $d_n^{(t+1)}$ and its associated edge should be removed. The figure also introduces the notation that we adopt for the different variables that serve to parameterize the messages. For Bernoulli message pdfs, we show only the non-zero probability, e.g., $\vec{\lambda}_n^{(t)} = \nu_{h_n^{(t)} \to s_n^{(t)}}(s_n^{(t)} = 1)$.

In the following subsections we will define the quantities that are shown in Fig. C.1, and illustrate how one can obtain estimates of $\{\boldsymbol{x}^{(t)}\}_{t=0}^T$. We use k to denote a DCS-AMP algorithmic iteration index. We primarily restrict our attention to the forward portion of the forward/backward pass, noting that most of the quantities can be straightforwardly obtained in the backward portion by a simple substitution of certain indices, (e.g., replacing $\overleftarrow{\lambda}_{nt}^{k-1}$ with $\overleftarrow{\lambda}_{nt}^{k}$). The notation k_f and k_b is used to distinguish between the k^{th} message on the *forward* portion of the forward/backward pass and the k^{th} message (if smoothing) on the *backward* portion of the pass. The reader may

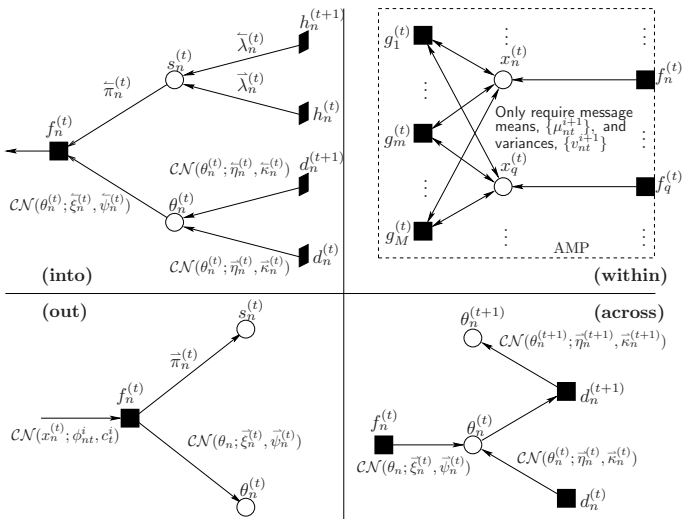


Figure C.1: A summary of the four message passing phases, including message notation and form.

find the following relation useful for the subsequent derivations: $\prod_q \mathcal{CN}(x;\mu_q,v_q) \propto \mathcal{CN}\left(x;\frac{\sum_q \mu_q/v_q}{\sum_q 1/v_q},\frac{1}{\sum_q 1/v_q}\right)$.

C.1 Derivation of (into) Messages

We begin by looking at the messages that are moving into a frame. First, we derive $\nu^{k_f}_{s_n^{(t)}\to f_n^{(t)}}(s_n^{(t)})$, which will define the message passing quantity $\overleftarrow{\pi}^{k_f}_{nt}$. For the case $0 \leq t \leq T-1$, obeying the sum-product update rules [55] gives

$$\begin{aligned}\nu^{k_f}_{s_n^{(t)}\to f_n^{(t)}}(s_n^{(t)}) &\propto \nu^{k}_{h_n^{(t)}\to s_n^{(t)}}(s_n^{(t)})\cdot \nu^{k-1}_{h_n^{(t+1)}\to s_n^{(t)}}(s_n^{(t)}) \\ &= \overrightarrow{\lambda}^{k}_{nt}\cdot\overleftarrow{\lambda}^{k-1}_{nt}.\end{aligned}$$

After appropriate scaling in order to yield a valid pmf, we obtain

$$\nu^{k_f}_{s_n^{(t)}\to f_n^{(t)}}(s_n^{(t)}) = \frac{\vec{\lambda}^k_{nt}\cdot\overleftarrow{\lambda}^{k-1}_{nt}}{\underbrace{(1-\vec{\lambda}^k_{nt})\cdot(1-\overleftarrow{\lambda}^{k-1}_{nt})+\vec{\lambda}^k_{nt}\cdot\overleftarrow{\lambda}^{k-1}_{nt}}_{\triangleq\overleftarrow{\pi}^{k_f}_{nt}}}. \tag{C.1}$$

For the case $t=T$, $\nu^{k_f}_{s_n^{(T)}\to f_n^{(T)}}(s_n^{(T)}) = \nu^{k}_{h_n^{(T)}\to s_n^{(T)}}(s_n^{(T)}) = \vec{\lambda}^k_{nT} \triangleq \overleftarrow{\pi}^{k_f}_{nT}$.

Next, we derive $\nu^{k_f}_{\theta_n^{(t)}\to f_n^{(t)}}(\theta_n^{(t)})$, which will define the quantities $\overleftarrow{\xi}^{k_f}_{nt}$ and $\overleftarrow{\psi}^{k_f}_{nt}$. For the case $0\le t\le T-1$,

$$\begin{aligned}
\nu^{k_f}_{\theta_n^{(t)}\to f_n^{(t)}}(\theta_n^{(t)}) &\propto \nu^{k}_{d_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)})\cdot\nu^{k-1}_{d_n^{(t+1)}\to\theta_n^{(t)}}(\theta_n^{(t)}) \\
&= \mathcal{CN}\left(\theta_n^{(t)};\vec{\eta}^k_{nt},\vec{\kappa}^k_{nt}\right)\cdot\mathcal{CN}\left(\theta_n^{(t)};\overleftarrow{\eta}^{k-1}_{nt},\overleftarrow{\kappa}^{k-1}_{nt}\right) \\
&\propto \mathcal{CN}\left(\theta_n^{(t)};\left(\frac{\vec{\kappa}^k_{nt}\cdot\overleftarrow{\kappa}^{k-1}_{nt}}{\vec{\kappa}^k_{nt}+\overleftarrow{\kappa}^{k-1}_{nt}}\right)\left(\frac{\vec{\eta}^k_{nt}}{\vec{\kappa}^k_{nt}}+\frac{\overleftarrow{\eta}^{k-1}_{nt}}{\overleftarrow{\kappa}^{k-1}_{nt}}\right),\left(\frac{\vec{\kappa}^k_{nt}\cdot\overleftarrow{\kappa}^{k-1}_{nt}}{\vec{\kappa}^k_{nt}+\overleftarrow{\kappa}^{k-1}_{nt}}\right)\right) \\
&= \mathcal{CN}\left(\theta_n^{(t)};\overleftarrow{\xi}^{k_f}_{nt},\overleftarrow{\psi}^{k_f}_{nt}\right),
\end{aligned} \tag{C.2}$$

where

$$\overleftarrow{\psi}^{k_f}_{nt} \triangleq \frac{\vec{\kappa}^k_{nt}\cdot\overleftarrow{\kappa}^{k-1}_{nt}}{\vec{\kappa}^k_{nt}+\overleftarrow{\kappa}^{k-1}_{nt}}, \tag{C.3}$$

$$\overleftarrow{\xi}^{k_f}_{nt} \triangleq \overleftarrow{\psi}^{k_f}_{nt}\cdot\left(\frac{\vec{\eta}^k_{nt}}{\vec{\kappa}^k_{nt}}+\frac{\overleftarrow{\eta}^{k-1}_{nt}}{\overleftarrow{\kappa}^{k-1}_{nt}}\right). \tag{C.4}$$

For the case $t=T$, $\nu^{k_f}_{\theta_n^{(T)}\to f_n^{(T)}}(\theta_n^{(T)}) = \mathcal{CN}(\theta_n^{(T)};\overleftarrow{\xi}^{k_f}_{nT},\overleftarrow{\psi}^{k_f}_{nT})$, where $\overleftarrow{\xi}^{k_f}_{nT} \triangleq \vec{\eta}^k_{nT}$, and $\overleftarrow{\psi}^{k_f}_{nT} \triangleq \vec{\kappa}^k_{nT}$.

Lastly, we derive the message $\nu^{k_f}_{f_n^{(t)}\to x_n^{(t)}}(x_n^{(t)})$, which sets the "local prior" for the

next execution of the AMP algorithm. Following the sum-product message computation rules,

$$\begin{aligned}
\nu^{k_f}_{f_n^{(t)}\to x_n^{(t)}}(x_n^{(t)}) &\propto \sum_{s_n^{(t)}=\{0,1\}} \int_{\theta_n^{(t)}} f_n^{(t)}(x_n^{(t)}, s_n^{(t)}, \theta_n^{(t)}) \cdot \nu^{k_f}_{s_n^{(t)}\to f_n^{(t)}}(s_n^{(t)}) \cdot \nu^{k_f}_{\theta_n^{(t)}\to f_n^{(t)}}(\theta_n^{(t)}) \\
&= \nu^{k_f}_{s_n^{(t)}\to f_n^{(t)}}(0) \int_{\theta_n^{(t)}} \delta(x_n^{(t)}) \cdot \nu^{k_f}_{\theta_n^{(t)}\to f_n^{(t)}}(\theta_n^{(t)}) d\theta_n^{(t)} \\
&\quad + \nu^{k_f}_{s_n^{(t)}\to f_n^{(t)}}(1) \int_{\theta_n^{(t)}} \delta(x_n^{(t)} - \theta_n^{(t)}) \cdot \nu^{k_f}_{\theta_n^{(t)}\to f_n^{(t)}}(\theta_n^{(t)}) d\theta_n^{(t)} \\
&= (1 - \overleftarrow{\pi}^{k_f}_{nt}) \int_u \delta(x_n^{(t)}) \cdot \mathcal{CN}(\theta_n^{(t)}; \overleftarrow{\xi}^{k_f}_{nt}, \overleftarrow{\psi}^{k_f}_{nt}) d\theta_n^{(t)} \\
&\quad + \overleftarrow{\pi}^{k_f}_{nt} \int_{\theta_n^{(t)}} \delta(x_n^{(t)} - \theta_n^{(t)}) \cdot \mathcal{CN}(\theta_n^{(t)}; \overleftarrow{\xi}^{k_f}_{nt}, \overleftarrow{\psi}^{k_f}_{nt}) d\theta_n^{(t)} \\
&= (1 - \overleftarrow{\pi}^{k_f}_{nt}) \delta(x_n^{(t)}) + \overleftarrow{\pi}^{k_f}_{nt} \mathcal{CN}(x_n^{(t)}; \overleftarrow{\xi}^{k_f}_{nt}, \overleftarrow{\psi}^{k_f}_{nt})
\end{aligned} \tag{C.5}$$

C.2 Derivation of (within) Messages

Since we are now focusing our attention only on messages passing within a single frame, and since these messages only depend on quantities that have been defined within that frame, in this subsection we will drop both the timestep indexing, t, and the forward/backward pass iteration indexing, k_f (or k_b), in order to simplify notation. Thus $\overleftarrow{\pi}^{k_f}_{nt}$, $\overleftarrow{\xi}^{k_f}_{nt}$, and $\overleftarrow{\psi}^{k_f}_{nt}$ become $\overleftarrow{\pi}_n$, $\overleftarrow{\xi}_n$, and $\overleftarrow{\psi}_n$ respectively. Likewise, $f_n^{(t)}$, $x_n^{(t)}$, $g_m^{(t)}$, $y_m^{(t)}$, and $\boldsymbol{A}^{(t)}$ become f_n, x_n, g_m, y_m, and $\boldsymbol{A}$, respectively. New variables defined in this section also retain an implicit dependence on both t and k_f/k_b. Additionally, we introduce a new index, i, which will serve to keep track of the multiple iterations of messages that pass back and forth between the $\{x_n^{(t)}\}$ and $\{g_m^{(t)}\}$ nodes within a frame during a single forward/backward pass.

Exact evaluation of $\nu^i_{x_n\to g_m}(x_n)$ and $\nu^i_{g_m\to x_n}(x_n)$ according to the rules of the standard sum-product algorithm would require the evaluation of many multi-dimensional

non-Gaussian integrals, which, for any appreciable size problem, quickly becomes intractable. Prior to introducing the AMP formalism, we first derive approximate belief propagation (BP) messages. These approximate BP messages are motivated by an observation that, for a sufficiently dense factor graph, the many non-Gaussian messages that arrive at a given g_m node yield, upon marginalizing according to the sum-product update rules, a message that can be well-approximated by a Gaussian (by appealing to central limit theorem arguments). It turns out that, if $\nu^i_{g_m \to x_n}(x_n)$ is approximately Gaussian, we do not need to know the precise form of $\nu^i_{x_n \to g_m}(x_n)$. Rather, it suffices to know just the mean and variance of the distribution. Let $\mu^i_{mn} \triangleq \int_{x_n} x_n \nu^i_{x_n \to g_m}(x_n) dx_n$ denote the mean, and $v^i_{mn} \triangleq \int_{x_n} |x_n - \mu^i_{mn}|^2 \nu^i_{x_n \to g_m}(x_n) dx_n$ denote the variance. Under the assumption of Gaussianity for $\nu^i_{g_m \to x_n}(x_n)$, it can be shown (see, e.g., [53]) that the sum-product update rules imply that

$$\nu^i_{g_m \to x_n}(x_n) = \mathcal{CN}\left(x_n; \frac{z^i_{mn}}{A_{mn}}, \frac{c^i_{mn}}{|A_{mn}|^2}\right), \tag{C.6}$$

$$z^i_{mn} \triangleq y_m - \sum_{q \neq n} A_{mq} \mu^i_{mq}, \tag{C.7}$$

$$c^i_{mn} \triangleq \sigma_e^2 + \sum_{q \neq n} |A_{mq}|^2 v^i_{mq}, \tag{C.8}$$

where A_{mn} refers to the $(m,n)^{th}$ element of $\boldsymbol{A}$. In order to compute μ^{i+1}_{mn} and v^{i+1}_{mn}, that is, the mean and variance of the message $\nu^{i+1}_{x_n \to g_m}(x_n)$, we must determine the mean and variance of the right-hand-side of the equation

$$\nu^{i+1}_{x_n \to g_m}(x_n) \propto \nu_{f_n \to x_n}(x_n) \prod_{l \neq m} \nu^i_{g_l \to x_n}(x_n). \tag{C.9}$$

Inserting (C.5) and (C.6) into (C.9) yields

$$\begin{aligned}\nu^{i+1}_{x_n\to g_m}(x_n) \quad &\propto \quad \left[(1-\overleftarrow{\pi}_n)\,\delta(x_n)+\overleftarrow{\pi}_n\,\mathcal{CN}(x_n;\overleftarrow{\xi}_n,\overleftarrow{\psi}_n)\right] \\ &\quad \times\mathcal{CN}\left(x_n;\frac{\sum_{l\neq m}A^*_{ln}z^i_{ln}/c^i_{ln}}{\sum_{l\neq m}|A_{ln}|^2/c^i_{ln}},\frac{1}{\sum_{l\neq m}|A_{ln}|^2/c^i_{ln}}\right). \qquad \text{(C.10)}\end{aligned}$$

In the large-system limit ($M,N\to\infty$ with M/N fixed), $c^i_{ln}\approx c^i_n\triangleq\frac{1}{M}\sum_{m=1}^M c^i_{mn}$, and, since the columns of $\boldsymbol{A}$ are of unit-norm, $\sum_{l\neq m}|A_{ln}|^2\approx 1$. Consequently, (C.10) simplifies to

$$\begin{aligned}\nu^{i+1}_{x_n\to g_m}(x_n) \quad &\propto \quad \left[(1-\overleftarrow{\pi}_n)\,\delta(x_n)+\overleftarrow{\pi}_n\,\mathcal{CN}(x_n;\overleftarrow{\xi}_n,\overleftarrow{\psi}_n)\right] \\ &\quad \times\mathcal{CN}\left(x_n;\textstyle\sum_{l\neq m}A^*_{ln}z^i_{ln},c^i_n\right). \qquad \text{(C.11)}\end{aligned}$$

Now, if we define

$$\phi^i_{mn} \triangleq \sum_{l\neq m}A^*_{ln}z^i_{ln}, \qquad \text{(C.12)}$$

$$\gamma^i_{mn} \triangleq \frac{(1-\overleftarrow{\pi}_n)\,\mathcal{CN}\left(0;\phi^i_{mn},c^i_n\right)}{\overleftarrow{\pi}_n\,\mathcal{CN}\left(0;\phi^i_{mn}-\overleftarrow{\xi}_n,c^i_n+\overleftarrow{\psi}_n\right)}, \qquad \text{(C.13)}$$

$$\bar{v}^i_n \triangleq \frac{c^i_n\overleftarrow{\psi}_n}{c^i_n+\overleftarrow{\psi}_n}, \qquad \text{(C.14)}$$

$$\bar{\mu}^i_{mn} \triangleq \bar{v}^i_n\left(\frac{\phi^i_{mn}}{c^i_n}+\frac{\overleftarrow{\xi}_n}{\overleftarrow{\psi}_n}\right), \qquad \text{(C.15)}$$

then (C.11) can be rewritten (after appropriate normalization to yield a valid pdf) as

$$\nu^{i+1}_{x_n\to g_m}(x_n)=\left(\frac{\gamma^i_{mn}}{1+\gamma^i_{mn}}\right)\delta(x_n)+\left(\frac{1}{1+\gamma^i_{mn}}\right)\mathcal{CN}(x_n;\bar{\mu}^i_{mn},\bar{v}^i_n). \qquad \text{(C.16)}$$

Equation (C.16) represents a Bernoulli-Gaussian pdf. The mean, μ^{i+1}_{mn}, and variance, v^{i+1}_{mn}, are therefore the mean and variance of a random variable distributed according

to (C.16), namely

$$\mu_{mn}^{i+1} = \frac{\bar{\mu}_{mn}^{i}}{1+\gamma_{mn}^{i}} \tag{C.17}$$

$$v_{mn}^{i+1} = \frac{\bar{v}_{n}^{i}}{1+\gamma_{mn}^{i}} + \gamma_{mn}^{i}|\mu_{mn}^{i+1}|^2. \tag{C.18}$$

C.3 Derivation of Signal MMSE Estimates

Once again, we drop the t and k_f (or k_b) indices in what follows.

A minimum mean squared error (MMSE) estimate of a coefficient x_n is given by the mean of its posterior distribution, $p(x_n|\boldsymbol{y})$. Additionally, the variance of the posterior distribution characterizes the MSE. In the BP framework, the posterior distribution of any variable (represented by a variable node in the factor graph) is given by the product of all incoming messages to the variable node. This implies that $p(x_n|\boldsymbol{y})$ is approximated at iteration $i+1$ by

$$\hat{p}^{i+1}(x_n|\boldsymbol{y}) \propto \nu_{f_n \to x_n}(x_n) \prod_{m=1}^{M} \nu_{g_m \to x_n}^{i}(x_n). \tag{C.19}$$

Careful examination of (C.19) reveals that it only differs from (C.9) by the inclusion of the m^{th} product term. Accounting for this additional term in a manner similar to that of Section C.2 results in the following MMSE expressions:

$$\mu_{n}^{i+1} \triangleq \hat{\mathrm{E}}^{i+1}[x_n|\boldsymbol{y}] = \frac{\bar{\mu}_{n}^{i}}{1+\gamma_{n}^{i}} \tag{C.20}$$

$$v_{n}^{i+1} \triangleq \widehat{\mathrm{var}}^{i+1}\{x_n|\boldsymbol{y}\} = \frac{\bar{v}_{n}^{i}}{1+\gamma_{n}^{i}} + \gamma_{n}^{i}|\mu_{n}^{i+1}|^2, \tag{C.21}$$

where γ_n^i and $\bar{\mu}_n^i$ are obtained straightforwardly by replacing ϕ_{mn}^i in (C.13), and (C.15) by $\phi_n^i \triangleq \sum_{m=1}^{M} A_{mn}^* z_{mn}^i$.

C.4 Derivation of AMP update equations

Here also we drop the t and k_f (or k_b) indices.

In many large-scale problems, it may be infeasible to track the $\mathcal{O}(MN)$ variables necessary to implement the message passes of Section C.2. In such cases, the *approximate message passing* (AMP) technique proposed by Donoho, Maleki, and Montanari offers an attractive alternative. AMP, like BP, is not a single algorithm, but rather a framework for constructing algorithms tailored to specific problem setups. By making a few key assumptions about the nature of the BP messages, the validity of which can be checked empirically, AMP is able to reduce the number of messages that must be tracked to $\mathcal{O}(N)$, resulting in algorithms which offer both high performance and computational efficiency. In this subsection we provide the update equations necessary to implement an AMP algorithm that serves as a substitute for the loopy BP method of Section C.2 and Section C.3.

Common to any AMP algorithm are the generic update equations [26] given by

$$\begin{aligned}
\phi_n^i &= \sum_{m=1}^{M} A_{mn}^* z_m^i + \mu_n^i, && \text{(C.22)}\\
\mu_n^{i+1} &= F_n(\phi_n^i; c^i), && \text{(C.23)}\\
v_n^{i+1} &= G_n(\phi_n^i; c^i), && \text{(C.24)}\\
c^{i+1} &= \sigma_e^2 + \tfrac{1}{M} \sum_{n=1}^{N} v_n^{i+1}, && \text{(C.25)}\\
z_m^{i+1} &= y_m - \sum_{n=1}^{N} A_{mn} \mu_n^{i+1} + \tfrac{z_m^i}{M} \sum_{n=1}^{N} F_n'(\phi_n^i; c^i). && \text{(C.26)}
\end{aligned}$$

The functions $F_n(\phi; c)$, $G_n(\phi; c)$, and $F_n'(\phi; c)$ that appear in (C.23), (C.24), and (C.26) are unique to the particular "local prior" under which we are operating AMP (see [26, §5] for definitions of F and G). Recalling the Bernoulli-Gaussian form of this

prior from (C.5), it can be shown that these functions are given by

$$F_n(\phi;c) \triangleq (1+\gamma_n(\phi;c))^{-1}\Big(\frac{\overleftarrow{\psi}_n\phi+\overleftarrow{\xi}_n c}{\overleftarrow{\psi}_n+c}\Big), \tag{C.27}$$

$$G_n(\phi;c) \triangleq (1+\gamma_n(\phi;c))^{-1}\Big(\frac{\overleftarrow{\psi}_n c}{\overleftarrow{\psi}_n+c}\Big)+\gamma_n(\phi;c)|F_n(\phi;c)|^2, \tag{C.28}$$

$$F'_n(\phi;c) \triangleq \tfrac{\partial}{\partial\phi}F_n(\phi,c)=\tfrac{1}{c}G_n(\phi;c), \tag{C.29}$$

where

$$\gamma_n(\phi;c)\triangleq\Big(\frac{1-\overleftarrow{\pi}_n}{\overleftarrow{\pi}_n}\Big)\Big(\frac{\overleftarrow{\psi}_n+c}{c}\Big)\exp\Big(-\Big[\frac{\overleftarrow{\psi}_n|\phi|^2+\overleftarrow{\xi}_n^* c\phi+\overleftarrow{\xi}_n c\phi^*-c|\overleftarrow{\xi}_n|^2}{c(\overleftarrow{\psi}_n+c)}\Big]\Big). \tag{C.30}$$

Note the strong similarity between (C.27) and (C.17), and between (C.28) and (C.18).

C.5 Derivation of (out) Messages

After passing a certain number of messages, (call that number I), between the $\{x_n^{(t)}\}$ and $\{g_m^{(t)}\}$ nodes, it becomes time to start passing messages back out of frame t. We now transition back to making use of the t and k_f (or k_b) indices again, which will require that we re-express ϕ_n^i and c^i, (for $i=I$), as $\phi_{nt}^{k_f}$ and $c_{nt}^{k_f}$, in order to make explicit their dependence on the timestep and forward/backward pass iteration.

The outgoing message from $f_n^{(t)}$ to $s_n^{(t)}$ is obtained as

$$\begin{aligned}\nu^{k_f}_{f_n^{(t)}\to s_n^{(t)}}(s_n^{(t)}) &\propto \int_{x_n^{(t)}}\int_{\theta_n^{(t)}} f_n^{(t)}(x_n^{(t)},s_n^{(t)},\theta_n^{(t)})\cdot\nu^{k_f}_{x_n^{(t)}\to f_n^{(t)}}(x_n^{(t)})\cdot\nu^{k}_{\theta_n^{(t)}\to f_n^{(t)}}(\theta_n^{(t)}),\\ &= \int_{x_n^{(t)}}\int_{\theta_n^{(t)}}\delta(x_n^{(t)}-s_n^{(t)}\theta_n^{(t)})\cdot\mathcal{CN}(x_n^{(t)};\phi_{nt}^{k_f},c_{nt}^{k_f})\cdot\mathcal{CN}(\theta_n^{(t)};\overleftarrow{\xi}_{nt}^{k},\overleftarrow{\psi}_{nt}^{k}).\end{aligned}$$

Performing the integration yields

$$\begin{aligned}\nu^{k_f}_{f_n^{(t)}\to s_n^{(t)}}(0) &\propto \mathcal{CN}(0;\phi^{k_f}_{nt},c_t^{k_f}),\\ \nu^{k_f}_{f_n^{(t)}\to s_n^{(t)}}(1) &\propto \mathcal{CN}(0;\phi^{k_f}_{nt}-\overleftarrow{\xi}^k_{nt},c_t^{k_f}+\overleftarrow{\psi}^k_{nt}).\end{aligned}$$

After normalizing to obtain a valid pdf, we find

$$\begin{aligned}\nu^{k_f}_{f_n^{(t)}\to s_n^{(t)}}(1) &= \frac{\mathcal{CN}(0;\phi^{k_f}_{nt}-\overleftarrow{\xi}^k_{nt},c_t^{k_f}+\overleftarrow{\psi}^k_{nt})}{\mathcal{CN}(0;\phi^{k_f}_{nt},c_t^{k_f})+\mathcal{CN}(0;\phi^{k_f}_{nt}-\overleftarrow{\xi}^k_{nt},c_t^{k_f}+\overleftarrow{\psi}^k_{nt})}\\ &= \underbrace{\left(1+\left(\frac{\overleftarrow{\pi}_n^{(t)}}{1-\overleftarrow{\pi}_n^{(t)}}\right)\gamma_{nt}(\phi^{k_f}_{nt},c_t^{k_f})\right)^{-1}}_{\triangleq\overrightarrow{\pi}^{k_f}_{nt}}. \qquad \text{(C.31)}\end{aligned}$$

The outgoing message from $f_n^{(t)}$ to $\theta_n^{(t)}$ is found by evaluating

$$\begin{aligned}\nu^{k_f,\text{exact}}_{f_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)}) &\propto \sum_{s_n^{(t)}=\{0,1\}}\int_{x_n^{(t)}} f_n^{(t)}(x_n^{(t)},s_n^{(t)},\theta_n^{(t)})\cdot\nu^{k_f}_{s_n^{(t)}\to f_n^{(t)}}(s_n^{(t)})\cdot\nu^{k_f}_{x_n^{(t)}\to f_n^{(t)}}(x_n^{(t)})\\ &= \sum_{s_n^{(t)}=\{0,1\}}\int_{x_n^{(t)}}\delta(x_n^{(t)}-s_n^{(t)}\theta_n^{(t)})\cdot\nu^{k_f}_{s_n^{(t)}\to f_n^{(t)}}(s_n^{(t)})\cdot\mathcal{CN}(x_n^{(t)};\phi^{k_f}_{nt},c^{k_f}_{nt}),\\ &= (1-\overleftarrow{\pi}_n^{(t)})\mathcal{CN}(0;\phi^{k_f}_{nt},c_t^{k_f})+\overleftarrow{\pi}_n^{(t)}\mathcal{CN}(\theta_n^{(t)};\phi^{k_f}_{nt},c_t^{k_f}). \qquad \text{(C.32)}\end{aligned}$$

Unfortunately, the term $\mathcal{CN}(0;\phi^i_{nt},c^i_t)$ prevents us from normalizing $\nu^{\text{exact}}_{f_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)})$, as the former is constant with respect to $\theta_n^{(t)}$. Therefore, the distribution on $\theta_n^{(t)}$ represented by (C.32) is improper. To avoid an improper pdf, we modify how this message is derived by regarding our assumed signal model, in which $s_n^{(t)}\in\{0,1\}$, as a limiting case of the model with $s_n^{(t)}\in\{\varepsilon,1\}$ as $\varepsilon\to 0$. For any fixed positive ε, the resulting message $\nu_{f_n^{(t)}\to\theta_n^{(t)}}(\cdot)$ is proper, given by

$$\nu^{k_f,\text{mod}}_{f_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)}) = (1-\Omega(\overleftarrow{\pi}_n^{(t)}))\ \mathcal{CN}(\theta_n^{(t)};\tfrac{1}{\varepsilon}\phi^{k_f}_{nt},\tfrac{1}{\varepsilon^2}c_t^{k_f})+\Omega(\overleftarrow{\pi}_n^{(t)})\ \mathcal{CN}(\theta_n^{(t)};\phi^{k_f}_{nt},c_t^{k_f}), \quad \text{(C.33)}$$

where

$$\Omega(\pi) \triangleq \frac{\varepsilon^2 \pi}{(1-\pi)+\varepsilon^2 \pi}. \tag{C.34}$$

The pdf in (C.33) is that of a binary Gaussian mixture. If we consider $\varepsilon \ll 1$, the first mixture component is extremely broad, while the second is more "informative," with mean $\phi_{nt}^{k_f}$ and variance $c_t^{k_f}$. The relative weight assigned to each component Gaussian is determined by the term $\Omega(\overleftarrow{\pi}_n^{(t)})$. Notice that the limit of this weighting term is the simple indicator function

$$\lim_{\varepsilon \to 0} \Omega(\pi) = \begin{cases} 0 & \text{if } 0 \leq \pi < 1, \\ 1 & \text{if } \pi = 1. \end{cases} \tag{C.35}$$

Since we cannot set $\varepsilon = 0$, we instead fix a small positive value, e.g., $\varepsilon = 10^{-7}$. In this case, (C.33) could then be used as the outgoing message. However, this presents a further difficulty: propagating a binary Gaussian mixture forward in time would lead to an exponenial growth in the number of mixture components at subsequent timesteps. To avoid the exponential growth in the number of mixture components, we collapse our binary Gaussian mixture to a single Gaussian component. This can be justified by the fact that, for $\varepsilon \ll 1$, $\Omega(\cdot)$ behaves nearly like the indicator function in (C.35), in which case one of the two Gaussian components will typically have negligible mass.

To carry out the Gaussian sum approximation, we propose choosing a threshold τ that is slightly smaller than 1 and, using (C.35) as a guide, threshold $\overleftarrow{\pi}_n^{(t)}$ to choose between the two Gaussian components of (C.33). The resultant message is thus

$$\nu_{f_n^{(t)} \to \theta_n^{(t)}}^{k_f}(\theta_n^{(t)}) = \mathcal{CN}(\theta_n^{(t)}; \vec{\xi}_n^{(t)}, \vec{\psi}_n^{(t)}), \tag{C.36}$$

with $\vec{\xi}_n^{(t)}$ and $\vec{\psi}_n^{(t)}$ chosen according to

$$\left(\vec{\xi}_n^{(t)}, \vec{\psi}_n^{(t)}\right) = \begin{cases} \left(\frac{1}{\varepsilon}\phi_{nt}^{k_f}, \frac{1}{\varepsilon^2}c_t^{k_f}\right), & \overleftarrow{\pi}_n^{(t)} \leq \tau \\ \left(\phi_{nt}^{k_f}, c_t^{k_f}\right), & \overleftarrow{\pi}_n^{(t)} > \tau \end{cases} . \tag{C.37}$$

C.6 Derivation of Forward-Propagating (across) Messages

We now consider forward-propagating inter-frame messages, i.e. those messages that move out of the frame of the current timestep and into the frame of the subsequent timestep. These messages are transmitted only during the forward portion of a forward/backward pass. First, we consider $\nu^k_{h_n^{(t+1)}\to s_n^{(t+1)}}(s_n^{(t+1)})$, the message that updates the prior on the signal support at the next timestep. For $t = 0, \ldots, T-1$, this message depends on outgoing messages from frame t. Specifically,

$$\begin{aligned} \nu^k_{h_n^{(t+1)}\to s_n^{(t+1)}}(s_n^{(t+1)}) &\propto \sum_{s_n^{(t)}=\{0,1\}} h_n^{(t+1)}(s_n^{(t+1)}, s_n^{(t)}) \cdot \nu^k_{s_n^{(t)}\to h_n^{(t+1)}}(s_n^{(t)}) \\ &\propto \sum_{s_n^{(t)}=\{0,1\}} p(s_n^{(t+1)}|s_n^{(t)}) \cdot \nu^k_{h_n^{(t)}\to s_n^{(t)}}(s_n^{(t)}) \cdot \nu^k_{f_n^{(t)}\to s_n^{(t)}}(s_n^{(t)}). \end{aligned}$$

Using the fact that $\nu^k_{h_n^{(t)}\to s_n^{(t)}}(1) = \vec{\lambda}^k_{nt}$ and $\nu^k_{f_n^{(t)}\to s_n^{(t)}}(1) = \vec{\pi}^{k_f}_{nt}$, it can be shown that

$$\nu^k_{h_n^{(t+1)}\to s_n^{(t+1)}}(1) = \underbrace{\frac{p_{10}(1-\vec{\lambda}^k_{nt})(1-\vec{\pi}^{k_f}_{nt}) + (1-p_{01})\vec{\lambda}^k_{nt}\vec{\pi}^{k_f}_{nt}}{(1-\vec{\lambda}^k_{nt})(1-\vec{\pi}^{k_f}_{nt}) + \vec{\lambda}^k_{nt}\vec{\pi}^{k_f}_{nt}}}_{\triangleq \vec{\lambda}^k_{n,t+1}} . \tag{C.38}$$

Note that $\vec{\lambda}^k_{n0} = \vec{\lambda}_{n0}$ for all k. In other words, $\nu^k_{h_n^{(0)}\to s_n^{(0)}}(1) = \vec{\lambda}_{n0}$ for each forward/backward pass, where $\vec{\lambda}_{n0}$ is the prior at timestep $t = 0$, i.e., $\vec{\lambda}_{n0} = \lambda$.

The other forward-propagating inter-frame message we need to characterize is $\nu^k_{d_n^{(t+1)}\to\theta_n^{(t+1)}}(\theta_n^{(t+1)})$, the message that updates the prior on active coefficient amplitudes at the next timestep. For $t = 0, \ldots, T-1$, BP update rules indicate that this

message is given as

$$\begin{aligned}
\nu^k_{d_n^{(t+1)}\to\theta_n^{(t+1)}}(\theta_n^{(t+1)}) &\propto \int_{\theta_n^{(t)}} d_n^{(t+1)}(\theta_n^{(t+1)},\theta_n^{(t)})\cdot \nu^k_{\theta_n^{(t)}\to d_n^{(t+1)}}(\theta_n^{(t)}) \\
&= \int_{\theta_n^{(t)}} p(\theta_n^{(t+1)}|\theta_n^{(t)})\cdot \nu^k_{d_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)})\cdot \nu^{k_f}_{f_n^{(t)}\to\theta_n^{(t)}}(\theta_n^{(t)}) \\
&= \int_{\theta_n^{(t)}} \mathcal{CN}(\theta_n^{(t+1)};(1-\alpha)\theta_n^{(t)}+\alpha\zeta,\alpha^2\rho)\cdot \mathcal{CN}(\theta_n^{(t)};\vec{\eta}^k_{nt},\vec{\kappa}^k_{nt})\cdot \\
&\qquad \mathcal{CN}(\theta_n^{(t)};\vec{\xi}^{k_f}_{nt},\vec{\psi}^{k_f}_{nt}).
\end{aligned}$$

Performing this integration, one finds

$$\nu^k_{d_n^{(t+1)}\to\theta_n^{(t+1)}}(\theta_n^{(t+1)}) = \mathcal{CN}(\theta_n^{(t+1)};\vec{\eta}^k_{nt},\vec{\kappa}^k_{nt}), \tag{C.39}$$

where

$$\vec{\eta}^k_{nt} \triangleq (1-\alpha)\left(\frac{\vec{\kappa}^k_{nt}\vec{\psi}^{k_f}_{nt}}{\vec{\kappa}^k_{nt}+\vec{\psi}^{k_f}_{nt}}\right)\left(\frac{\vec{\eta}^k_{nt}}{\vec{\kappa}^k_{nt}}+\frac{\vec{\xi}^{k_f}_{nt}}{\vec{\psi}^{k_f}_{nt}}\right)+\alpha\zeta, \tag{C.40}$$

$$\vec{\kappa}^k_{nt} \triangleq (1-\alpha)^2\left(\frac{\vec{\kappa}^k_{nt}\vec{\psi}^{k_f}_{nt}}{\vec{\kappa}^k_{nt}+\vec{\psi}^{k_f}_{nt}}\right)+\alpha^2\rho. \tag{C.41}$$

For the special case $t=1$, $\nu^k_{d_n^{(1)}\to\theta_n^{(1)}}(\theta_n^{(1)}) = p(\theta_n^{(1)}) = \mathcal{CN}(\theta_n^{(1)};\zeta,\sigma^2)$, thus $\vec{\eta}^k_{n1}=\zeta$ and $\vec{\kappa}^k_{n1}=\sigma^2$ for all k.

C.7 Derivation of Backward-Propagating (across) Messages

The final messages that we need to characterize are the backward-propagating inter-frame messages. These are the messages that, at the current timestep, are used to update the priors for the earlier timestep. Using analysis similar to that of Section C.6, one can verify that, for $t=2,\ldots,T-1$, the appropriate message updates

are given by

$$\nu^k_{h_n^{(t)}\to s_n^{(t-1)}}(1) = \overleftarrow{\lambda}^k_{n,t-1}, \tag{C.42}$$

$$\nu^k_{d_n^{(t)}\to \theta_n^{(t-1)}}(\theta_n^{(t-1)}) = \mathcal{CN}(\theta_n^{(t-1)}; \overleftarrow{\eta}^k_{nt}, \overleftarrow{\kappa}^k_{nt}), \tag{C.43}$$

where

$$\overleftarrow{\lambda}^k_{n,t-1} \triangleq \frac{p_{01}(1-\overleftarrow{\lambda}^k_{nt})(1-\vec{\pi}^{k_b}_{nt}) + (1-p_{01})\overleftarrow{\lambda}^k_{nt}\vec{\pi}^{k_b}_{nt}}{(1-p_{10}+p_{01})(1-\overleftarrow{\lambda}^k_{nt})(1-\vec{\pi}^{k_b}_{nt}) + (1-p_{01}+p_{10})\overleftarrow{\lambda}^k_{nt}\vec{\pi}^{k_b}_{nt}}, \tag{C.44}$$

$$\overleftarrow{\eta}^k_{n,t-1} \triangleq \frac{1}{(1-\alpha)}\left(\frac{\overleftarrow{\kappa}^k_{nt}\vec{\psi}^{k_b}_{nt}}{\overleftarrow{\kappa}^k_{nt}+\vec{\psi}^{k_b}_{nt}}\right)\left(\frac{\overleftarrow{\eta}^k_{nt}}{\overleftarrow{\kappa}^k_{nt}} + \frac{\vec{\xi}^{k_b}_{nt}}{\vec{\psi}^{k_b}_{nt}}\right) - \alpha\zeta, \tag{C.45}$$

$$\overleftarrow{\kappa}^k_{n,t-1} \triangleq \frac{1}{(1-\alpha)^2}\left[\left(\frac{\overleftarrow{\kappa}^k_{nt}\vec{\psi}^{k_b}_{nt}}{\overleftarrow{\kappa}^k_{nt}+\vec{\psi}^{k_b}_{nt}}\right) + \alpha^2\rho\right]. \tag{C.46}$$

For the special case $t = T$, the quantities $\overleftarrow{\lambda}^k_{n,T-1}$, $\overleftarrow{\eta}^k_{n,T-1}$, and $\overleftarrow{\kappa}^k_{n,T-1}$ can be obtained using (C.44)-(C.46) with the substitutions $\overleftarrow{\lambda}^k_{nT} = \frac{1}{2}$, $\overleftarrow{\eta}^k_{nT} = 0$, and $\overleftarrow{\kappa}^k_{nT} = \infty$.

Appendix D

DCS-AMP EM UPDATE DERIVATIONS

In this appendix, we provide derivations of the expectation-maximization (EM) learning update expressions that are used to automatically tune the model parameters of the DCS-AMP signal model described in Section 3.2. We assume some familiarity with the EM algorithm [65]. Those looking for a helpful tutorial on the basics of the EM algorithm may find [128] beneficial.

Let $\Gamma \triangleq \{\lambda, p_{01}, \zeta, \alpha, \rho, \sigma_e^2\}$ denote the set of all model parameters, and let Γ^k denote the set of parameter estimates at the k^{th} EM iteration. The objective of the EM procedure is to find parameter estimates that maximize the data likelihood $p(\bar{\boldsymbol{y}}|\Gamma)$. Since it is often computationally intractable to perform this maximization, the EM algorithm incorporates additional "hidden" data and iterates between two steps: *(i)* evaluating the conditional expectation of the log likelihood of the hidden data given the observed data, $\bar{\boldsymbol{y}}$, and the current estimates of the parameters, Γ^k, and *(ii)* maximizing this expected log likelihood with respect to the model parameters. For all parameters except σ_e^2 we use $\boldsymbol{s}$ and $\bar{\boldsymbol{\theta}}$ as the hidden data, while for σ_e^2 we use $\bar{\boldsymbol{x}}$.

Recall that the sum-product incarnation of belief propagation provides marginal, and pairwise joint, posterior distributions for all random variables in the factor graph [67]. Therefore, in the following derivations, we will leave the final updates expressed in terms of relevant moments of these distributions. In what follows, we

Distribution	Functional Form
$p\big(y_m^{(t)}\|\boldsymbol{x}^{(t)}\big)$	$\mathcal{CN}\big(y_m^{(t)}; \boldsymbol{a}_m^{(t)\,T}\boldsymbol{x}^{(t)}, \sigma_e^2\big)$
$p\big(x_n^{(t)}\|s_n^{(t)}, \theta_n^{(t)}\big)$	$\delta\big(x_n^{(t)} - s_n^{(t)}\theta_n^{(t)}\big)$
$p\big(s_n^{(1)}\big)$	$\big(1-\lambda\big)^{1-s_n^{(1)}}\lambda^{s_n^{(1)}}$
$p\big(s_n^{(t)}\|s_n^{(t-1)}\big)$	$\begin{cases} (1-p_{10})^{1-s_n^{(t)}}{p_{10}}^{s_n^{(t)}}, & s_n^{(t-1)}=0 \\ {p_{01}}^{1-s_n^{(t)}}(1-p_{01})^{s_n^{(t)}}, & s_n^{(t-1)}=1 \end{cases}$
$p\big(\theta_n^{(1)}\big)$	$\mathcal{CN}\big(\theta_n^{(1)}; \zeta, \sigma^2\big)$
$p\big(\theta_n^{(t)}\|\theta_n^{(t-1)}\big)$	$\mathcal{CN}\big(\theta_n^{(t)}; (1-\alpha)\theta_n^{(t-1)} + \alpha\zeta, \alpha^2\rho\big)$

Table D.1: The underlying distributions, and functional forms associated with the DCS-AMP signal model

define $Q_{\mathcal{H}}(\beta;\Gamma^k)$ as the conditional (on hidden data, $\mathcal{H}$) likelihood that is evaluated by the EM algorithm under parameter estimates Γ^k, as a function of β, a parameter being optimized, e.g.,

$$Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(\beta;\Gamma^k) \triangleq \mathrm{E}_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}\big[\log p\big(\bar{\boldsymbol{y}},\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}};\lambda,\Gamma^k\setminus\{\beta^k\}\big)\big|\bar{\boldsymbol{y}};\Gamma^k\big]. \tag{D.1}$$

For convenience, in Table D.1, we summarize relevant distributions from the signal model of Section 3.2 that will be useful in computing the necessary EM updates.

D.1 Sparsity Rate Update: λ^{k+1}

Beginning with the EM algorithm objective, we have that

$$\lambda^{k+1} = \underset{\lambda}{\operatorname{argmax}}\, Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(\lambda;\Gamma^k). \tag{D.2}$$

To solve (D.2), we first differentiate $Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(\lambda;\Gamma^k)$ with respect to λ:

$$\begin{aligned}
\frac{\partial Q}{\partial \lambda} &= \frac{\partial}{\partial \lambda}\mathrm{E}_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}\big[\log p\big(\bar{\boldsymbol{y}},\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}};\lambda,\Gamma^k\setminus\{\lambda^k\}\big)\big|\bar{\boldsymbol{y}};\Gamma^k\big], \\
&= \frac{\partial}{\partial \lambda}\sum_{n=1}^{N}\mathrm{E}_{s_n^{(1)}|\bar{\boldsymbol{y}}}\big[\log p\big(s_n^{(1)}\big)\big|\bar{\boldsymbol{y}}\big], \\
&= \sum_{n=1}^{N}\mathrm{E}\bigg[\frac{\partial}{\partial \lambda}\log p\big(s_n^{(1)}\big)\bigg|\bar{\boldsymbol{y}}\bigg], \\
&= \sum_{n=1}^{N}\mathrm{E}\bigg[\frac{\partial}{\partial \lambda}(1-s_n^{(1)})\log(1-\lambda)+s_n^{(1)}\log\lambda\bigg|\bar{\boldsymbol{y}}\bigg], \\
&= \sum_{n=1}^{N}\mathrm{E}\bigg[\frac{s_n^{(1)}}{\lambda}-\frac{1-s_n^{(1)}}{1-\lambda}\bigg|\bar{\boldsymbol{y}}\bigg], \\
&= \sum_{n=1}^{N}\frac{1}{\lambda}\mathrm{E}\big[s_n^{(1)}\big|\bar{\boldsymbol{y}}\big]-\frac{1}{1-\lambda}\big(1-\mathrm{E}\big[s_n^{(1)}\big|\bar{\boldsymbol{y}}\big]\big). \qquad \text{(D.3)}
\end{aligned}$$

Setting (D.3) equal to zero and solving for λ yields the desired EM update for the sparsity rate:

$$\lambda^{k+1}=\frac{1}{N}\sum_{n=1}^{N}\mathrm{E}\big[s_n^{(1)}\big|\bar{\boldsymbol{y}}\big]. \qquad \text{(D.4)}$$

D.2 Markov Transition Probability Update: p_{01}^{k+1}

Proceeding in a fashion similar to that of Appendix D.1, the active-to-inactive Markov transition probability EM update is given by

$$p_{01}^{k+1}=\underset{p_{01}}{\operatorname{argmax}}\,Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(p_{01};\Gamma^k). \qquad \text{(D.5)}$$

Differentiating $Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(p_{01};\Gamma^k)$ w.r.t. p_{01} gives

$$\frac{\partial Q}{\partial p_{01}} = \sum_{t=2}^{T}\sum_{n=1}^{N}\mathrm{E}_{s_n^{(t)},s_n^{(t-1)}|\bar{\boldsymbol{y}}}\bigg[\frac{\partial}{\partial p_{01}}\log p\big(s_n^{(t)},s_n^{(t-1)}\big)\bigg|\bar{\boldsymbol{y}}\bigg],$$

$$
\begin{aligned}
&= \sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}\left[p_{01}^{-1}(1-s_n^{(t)})s_n^{(t-1)} - (1-p_{01})^{-1} s_n^{(t)} s_n^{(t-1)} \middle| \bar{\boldsymbol{y}} \right], \\
&= \sum_{t=2}^{T}\sum_{n=1}^{N} p_{01}^{-1}\left(\mathrm{E}\big[s_n^{(t-1)} \big| \bar{\boldsymbol{y}} \big] - \mathrm{E}\big[s_n^{(t)} s_n^{(t-1)} \big| \bar{\boldsymbol{y}} \big] \right) - \\
&\qquad (1-p_{01})^{-1} \mathrm{E}\big[s_n^{(t)} s_n^{(t-1)} \big| \bar{\boldsymbol{y}} \big]. \qquad \text{(D.6)}
\end{aligned}
$$

Setting (D.6) equal to zero and solving for p_{01} yields the desired EM update:

$$
p_{01}^{k+1} = \frac{\sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}\big[s_n^{(t-1)} \big| \bar{\boldsymbol{y}} \big] - \mathrm{E}\big[s_n^{(t)} s_n^{(t-1)} \big| \bar{\boldsymbol{y}} \big]}{\sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}\big[s_n^{(t-1)} \big| \bar{\boldsymbol{y}} \big]}. \qquad \text{(D.7)}
$$

D.3 Amplitude Mean Update: ζ^{k+1}

The mean of the Gauss-Markov amplitude evolution process can be updated via EM according to

$$
\zeta^{k+1} = \underset{\zeta}{\operatorname{argmax}}\, Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(\zeta; \Gamma^k). \qquad \text{(D.8)}
$$

Differentiating $Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(\zeta;\Gamma^k)$ w.r.t. ζ gives

$$
\begin{aligned}
\frac{\partial Q}{\partial \zeta} &= \sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}_{\theta_n^{(t)},\theta_n^{(t-1)}|\bar{\boldsymbol{y}}}\left[\frac{\partial}{\partial \zeta} \log p\big(\theta_n^{(t)}, \theta_n^{(t-1)}\big) \middle| \bar{\boldsymbol{y}} \right] + \sum_{n=1}^{N} \mathrm{E}_{\theta_n^{(1)}|\bar{\boldsymbol{y}}}\left[\frac{\partial}{\partial \zeta} \log p(\theta_n^{(1)}) \middle| \bar{\boldsymbol{y}} \right], \\
&= \sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}\left[\tfrac{1}{\alpha^k \rho^k}\big(\theta_n^{(t)} - (1-\alpha^k)\theta_n^{(t-1)}\big) - \tfrac{1}{\rho^k}\zeta \middle| \bar{\boldsymbol{y}} \right] + \sum_{n=1}^{N} \mathrm{E}\left[\tfrac{1}{(\sigma^2)^k}\big(\theta_n^{(1)} - \zeta\big) \middle| \bar{\boldsymbol{y}} \right], \\
&= \sum_{t=2}^{T}\sum_{n=1}^{N} \tfrac{1}{\alpha^k \rho^k}\big(\bar{\mu}_n^{(t)} - (1-\alpha^k)\bar{\mu}^{(t-1)}\big) - \tfrac{1}{\rho^k}\zeta + \sum_{n=1}^{N} \tfrac{1}{(\sigma^2)^k}\big(\bar{\mu}_n^{(1)} - \zeta\big), \qquad \text{(D.9)}
\end{aligned}
$$

where $\bar{\mu}_n^{(t)} \triangleq \mathrm{E}_{\theta_n^{(t)}|\boldsymbol{y}}[\theta_n^{(t)}|\boldsymbol{y}]$. Setting (D.9) equal to zero and solving for ζ gives a final update expression of

$$
\zeta^{k+1} = \frac{\sum_{t=2}^{T}\sum_{n=1}^{N} \frac{1}{\alpha^k\rho^k}(\bar{\mu}_n^{(t)} - (1-\alpha^k)\bar{\mu}_n^{(t-1)}) + \sum_{n=1}^{N} \bar{\mu}_n^{(1)}/(\sigma^2)^k}{N((T-1)/\rho^k + 1/(\sigma^2)^k)}. \qquad \text{(D.10)}
$$

D.4 Amplitude Correlation Update: α^{k+1}

The Gauss-Markov amplitude evolution process is controlled in part by correlation parameter α. Proceeding straightforwardly as before, the EM update is given by

$$\alpha^{k+1} = \underset{\alpha}{\operatorname{argmax}}\, Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(\alpha;\Gamma^k). \tag{D.11}$$

Differentiating $Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(\alpha;\Gamma^k)$ w.r.t. α gives

$$\begin{aligned}
\frac{\partial Q}{\partial \alpha} &= \sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}_{\theta_n^{(t)},\theta_n^{(t-1)}|\bar{\boldsymbol{y}}}\left[\frac{\partial}{\partial\alpha}\log p\big(\theta_n^{(t)},\theta_n^{(t-1)}\big)\,\middle|\,\bar{\boldsymbol{y}}\right] \\
&= \sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}\left[\frac{\partial}{\partial\alpha}\log \mathcal{CN}\big(\theta_n^{(t)};(1-\alpha)\theta_n^{(t-1)}+\alpha\zeta^k,\alpha^2\rho^k\big)\,\middle|\,\bar{\boldsymbol{y}}\right] \\
&= \sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}\left[\frac{\partial}{\partial\alpha}\left\{-\log(\alpha^2\rho^k)-\frac{1}{\alpha^2\rho^k}\big|\theta_n^{(t)}-(1-\alpha)\theta_n^{(t-1)}-\alpha\zeta^k\big|^2\right\}\,\middle|\,\bar{\boldsymbol{y}}\right] \\
&= \sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}\left[\frac{\partial}{\partial\alpha}\left\{-\frac{2}{\alpha}+\frac{2}{\alpha^3\rho^k}\big|\theta_n^{(t)}-(1-\alpha)\theta_n^{(t-1)}-\alpha\zeta^k\big|^2-\right.\right. \\
&\qquad \frac{1}{\alpha^2\rho^k}\Big(2\,\mathrm{Re}\{\theta_n^{(t)}\theta_n^{(t-1)^*}\}-2\,\mathrm{Re}\{\zeta^k\theta_n^{(t)^*}\}+2(\alpha-1)\big|\theta_n^{(t-1)}\big|^2+ \\
&\qquad \left.\left.2(1-2\alpha)\,\mathrm{Re}\{\zeta^k\theta_n^{(t-1)^*}\}+2\alpha\big|\zeta^k\big|^2\Big)\right\}\,\middle|\,\bar{\boldsymbol{y}}\right].
\end{aligned} \tag{D.12}$$

Setting (D.12) equal to zero, and multiplying both sides by α^3 yields

$$0 = -\mathfrak{a}\alpha^2+\mathfrak{b}\alpha+\mathfrak{c}, \tag{D.13}$$

where $\mathfrak{a} \triangleq 2N(T-1)$, $\mathfrak{b} \triangleq \frac{2}{\rho^k}\sum_{t=2}^{T}\sum_{n=1}^{N}\mathrm{Re}\left\{\mathrm{E}[\theta_n^{(t)^*}\theta_n^{(t-1)}|\bar{\boldsymbol{y}}]-(\bar{\mu}_n^{(t)}-\bar{\mu}_n^{(t-1)})^*\zeta^k\right\}-\bar{v}_n^{(t-1)}-|\bar{\mu}_n^{(t-1)}|^2$, and $\mathfrak{c} \triangleq \frac{2}{\rho^k}\sum_{t=2}^{T}\sum_{n=1}^{N}\bar{v}_n^{(t)}+|\bar{\mu}_n^{(t)}|^2+\bar{v}_n^{(t-1)}+|\bar{\mu}_n^{(t-1)}|^2-2\,\mathrm{Re}\left\{\mathrm{E}[\theta_n^{(t)^*}\theta_n^{(t-1)}|\bar{\boldsymbol{y}}]\right\}$, for $\bar{\mu}_n^{(t)}$ defined as in Appendix D.3, and $\bar{v}_n^{(t)} \triangleq \mathrm{var}\{\theta_n^{(t)}|\bar{\boldsymbol{y}}\}$. Equation (D.13) can be recognized as a quadratic equation, the positive, real root of which gives the desired

update for α, namely,

$$\alpha^{k+1} = \tfrac{1}{2\mathfrak{a}}\Big(\mathfrak{b} - \sqrt{\mathfrak{b}^2 + 4\mathfrak{a}\mathfrak{c}}\Big). \tag{D.14}$$

D.5 Perturbation Variance Update: ρ^{k+1}

The EM update of the Gauss-Markov amplitude perturbation variance, ρ, is given by

$$\rho^{k+1} = \underset{\rho}{\operatorname{argmax}}\, Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(\rho;\Gamma^k). \tag{D.15}$$

Differentiating $Q_{\bar{\boldsymbol{s}},\bar{\boldsymbol{\theta}}|\bar{\boldsymbol{y}}}(\rho;\Gamma^k)$ w.r.t. ρ gives

$$\begin{aligned}
\frac{\partial Q}{\partial \rho} &= \sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}_{\theta_n^{(t)},\theta_n^{(t-1)}|\bar{\boldsymbol{y}}}\left[\frac{\partial}{\partial\rho}\log p\big(\theta_n^{(t)},\theta_n^{(t-1)}\big)\,\middle|\,\bar{\boldsymbol{y}}\right] \\
&= \sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}\left[\frac{\partial}{\partial\rho}\log \mathcal{CN}\big(\theta_n^{(t)};(1-\alpha^k)\theta_n^{(t-1)}+\alpha^k\zeta^k,(\alpha^k)^2\rho\big)\,\middle|\,\bar{\boldsymbol{y}}\right] \\
&= \tfrac{-N(T-1)}{\rho} + \tfrac{1}{(\alpha^k)^2\rho^2}\sum_{t=2}^{T}\sum_{n=1}^{N} \mathrm{E}\left[\left|\theta_n^{(t)}-(1-\alpha^k)\theta_n^{(t-1)}-\alpha^k\zeta^k\right|^2\,\middle|\,\bar{\boldsymbol{y}}\right]. \qquad \text{(D.16)}
\end{aligned}$$

Setting (D.16) equal to zero, and multiplying both sides of the resultant equality by ρ^2 gives a final update of

$$\begin{aligned}
\rho^{k+1} = \tfrac{1}{N(T-1)(\alpha^k)^2}\sum_{t=2}^{T}\sum_{n=1}^{N} &\bar{v}_n^{(t)} + |\bar{\mu}_n^{(t)}|^2 + (\alpha^k)^2|\zeta^k|^2 - 2\alpha^k \operatorname{Re}\left\{\bar{\mu}_n^{(t)*}\zeta^k\right\} - \\
&2(1-\alpha^k)\operatorname{Re}\left\{\mathrm{E}[\theta_n^{(t)*}\theta_n^{(t-1)}|\bar{\boldsymbol{y}}]\right\} + 2\alpha^k(1-\alpha^k)\times \\
&\operatorname{Re}\left\{\bar{\mu}_n^{(t-1)*}\zeta^k\right\} + (1-\alpha^k)(\bar{v}_n^{(t-1)} + |\bar{\mu}_n^{(t-1)}|^2). \qquad \text{(D.17)}
\end{aligned}$$

D.6 Noise Variance Update: $(\sigma_e^2)^{k+1}$

The final parameter that we must update is the noise variance, σ_e^2. Using $\bar{\boldsymbol{x}}$ as the hidden data, we solve

$$(\sigma_e^2)^{k+1} = \underset{\sigma_e^2}{\operatorname{argmax}}\, Q_{\bar{\boldsymbol{x}}|\bar{\boldsymbol{y}}}(\sigma_e^2; \Gamma^k). \tag{D.18}$$

Differentiating $Q_{\bar{\boldsymbol{x}}|\bar{\boldsymbol{y}}}(\sigma_e^2; \Gamma^k)$ w.r.t. σ_e^2 gives

$$\begin{aligned}
\frac{\partial Q}{\partial \sigma_e^2} &= \sum_{t=1}^{T}\sum_{m=1}^{M} \mathrm{E}_{\bar{\boldsymbol{x}}|\bar{\boldsymbol{y}}}\left[\frac{\partial}{\partial \sigma_e^2}\log p(y_m^{(t)}|\boldsymbol{x}^{(t)})\middle|\bar{\boldsymbol{y}}\right] \\
&= \frac{-MT}{\sigma_e^2} + \frac{1}{(\sigma_e^2)^2}\sum_{t=1}^{T}\sum_{m=1}^{M} \mathrm{E}_{\bar{\boldsymbol{x}}|\bar{\boldsymbol{y}}}\left[\left|y_m^{(t)} - \sum_{n=1}^{N} a_{mn}^{(t)} x_n^{(t)}\right|^2\middle|\bar{\boldsymbol{y}}\right] \qquad \text{(D.19)} \\
&= \frac{-MT}{\sigma_e^2} + \frac{1}{(\sigma_e^2)^2}\sum_{t=1}^{T} \|\boldsymbol{y}^{(t)} - \boldsymbol{A}^{(t)}\boldsymbol{\mu}^{(t)}\|_2^2 + \mathbf{1}^{\mathsf{T}}\boldsymbol{v}^{(t)}, \qquad \text{(D.20)}
\end{aligned}$$

where $\mu_n^{(t)} \triangleq \mathrm{E}[x_n^{(t)}|\bar{\boldsymbol{y}}]$ and $v_n^{(t)} \triangleq \mathrm{var}\{x_n^{(t)}|\bar{\boldsymbol{y}}\}$. In moving from (D.19) to (D.20), we must assume pairwise posterior independence of the coefficients of $\boldsymbol{x}^{(t)}$, that is, $p(x_n^{(t)}, x_q^{(t)}|\bar{\boldsymbol{y}}) \approx p(x_n^{(t)}|\bar{\boldsymbol{y}})p(x_q^{(t)}|\bar{\boldsymbol{y}})$, which is a reasonable assumption for high-dimensional problems. Setting (D.20) equal to zero and solving for $(\sigma_e^2)^{k+1}$ gives

$$(\sigma_e^2)^{k+1} = \frac{1}{MT}\sum_{t=1}^{T} \|\boldsymbol{y}^{(t)} - \boldsymbol{A}^{(t)}\boldsymbol{\mu}^{(t)}\|_2^2 + \mathbf{1}^{\mathsf{T}}\boldsymbol{v}^{(t)}, \tag{D.21}$$

completing the DCS-AMP EM update derivations.

Appendix E

GAMP CLASSIFICATION DERIVATIONS

In this appendix, we provide derivations of a state evolution covariance matrix Σ_z^k, as well as GAMP message update equations for several likelihoods/activation functions described in Section 4.4, and EM parameter update procedures for the logistic and probit activation functions.

E.1 Derivation of Σ_z^k

Recall from Section 4.3 that

$$\begin{bmatrix} \mathsf{z} \\ \hat{\mathsf{z}}^k \end{bmatrix} \xrightarrow{d} \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_z^k) = \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma_{11}^k & \Sigma_{12}^k \\ \Sigma_{21}^k & \Sigma_{22}^k \end{bmatrix} \right). \tag{E.1}$$

In this section, we derive expressions for the components of Σ_z^k in terms of quantities that are tracked as part of GAMP's state evolution formalism [27], namely $\mathrm{E}[\mathsf{w}_n]$, $\mathrm{E}[\widehat{\mathsf{w}}_n^k]]$, $\mathrm{var}\{\mathsf{w}_n\}$, $\mathrm{var}\{\widehat{\mathsf{w}}_n^k\}$, and $\mathrm{cov}\{\mathsf{w}_n, \widehat{\mathsf{w}}_n^k\}$.

Beginning with Σ_{11}^k, from the definition of z, and the fact that $\mathrm{E}_{\mathsf{x}_n}[\mathsf{x}_n] = 0$ and $\mathrm{var}\{\mathsf{x}_n\} = 1/M$, we find that

$$\begin{aligned} \Sigma_{11}^k &\triangleq \mathrm{var}\{\mathsf{z}\} \\ &= \mathrm{E}_{\mathsf{z}}[\mathsf{z}^2] \end{aligned}$$

$$
\begin{aligned}
&= \mathrm{E}_{\mathsf{x},\mathsf{w}}\Big[\Big(\sum_n \mathsf{x}_n \mathsf{w}_n\Big)^2\Big] \\
&= \mathrm{E}_{\mathsf{x},\mathsf{w}}\Big[\sum_n \mathsf{x}_n^2 \mathsf{w}_n^2\Big] + 2\sum_n \sum_{q<n} \mathrm{E}_{\mathsf{x},\mathsf{w}}\Big[\mathsf{x}_n \mathsf{x}_q \mathsf{w}_n \mathsf{w}_q\Big] \\
&= \sum_n \mathrm{E}_{\mathsf{x}_n}[\mathsf{x}_n^2]\mathrm{E}_{\mathsf{w}_n}[\mathsf{w}_n^2] \\
&= \sum_n \mathrm{var}\{\mathsf{x}_n\}\big(\mathrm{var}\{\mathsf{w}_n\} + \mathrm{E}[\mathsf{w}_n]^2\big) \\
&= \delta^{-1}\big(\mathrm{var}\{\mathsf{w}_n\} + \mathrm{E}[\mathsf{w}_n]^2\big), \qquad \text{(E.2)}
\end{aligned}
$$

where $\delta \triangleq M/N$. By an analogous argument, it follows that

$$
\Sigma_{22}^k = \delta^{-1}\big(\mathrm{var}\{\hat{\mathsf{w}}_n^k\} + \mathrm{E}[\hat{\mathsf{w}}_n^k]^2\big). \qquad \text{(E.3)}
$$

Finally,

$$
\begin{aligned}
\Sigma_{12}^k &= \Sigma_{21}^k = \mathrm{cov}\{\mathsf{z}, \hat{\mathsf{z}}^k\} \\
&= \mathrm{E}_{\mathsf{z},\hat{\mathsf{z}}^k}[\mathsf{z}, \hat{\mathsf{z}}^k] \\
&= \mathrm{E}_{\mathsf{x},\mathsf{w},\hat{\mathsf{w}}^k}\Big[\Big(\sum_n \mathsf{x}_n \mathsf{w}_n\Big)\Big(\sum_q \mathsf{x}_q \hat{\mathsf{w}}_q^k\Big)\Big] \\
&= \sum_n \sum_q \mathrm{E}_{\mathsf{x},\mathsf{w}_n,\hat{\mathsf{w}}_n^k}[\mathsf{x}_n \mathsf{x}_q \mathsf{w}_n \hat{\mathsf{w}}_q^k] \\
&= \sum_n \mathrm{E}_{\mathsf{x}_n,\mathsf{w}_n,\hat{\mathsf{w}}_n^k}[\mathsf{x}_n^2 \mathsf{w}_n \hat{\mathsf{w}}_n^k] \\
&= \sum_n \mathrm{var}\{\mathsf{x}_n\}\mathrm{E}_{\mathsf{w}_n,\hat{\mathsf{w}}_n^k}[\mathsf{w}_n \hat{\mathsf{w}}_n^k] \\
&= \sum_n \mathrm{var}\{\mathsf{x}_n\}\big(\mathrm{cov}\{\mathsf{w}_n, \widehat{\mathsf{w}}_n^k\} + \mathrm{E}[\mathsf{w}_n]\mathrm{E}[\hat{\mathsf{w}}_n^k]\big) \\
&= \delta^{-1}\big(\mathrm{cov}\{\mathsf{w}_n, \widehat{\mathsf{w}}_n^k\} + \mathrm{E}[\mathsf{w}_n]\mathrm{E}[\hat{\mathsf{w}}_n^k]\big). \qquad \text{(E.4)}
\end{aligned}
$$

E.2 Sum-Product GAMP Updates for a Logistic Likelihood

In this section, we describe a variational inference technique for approximating the sum-product GAMP updates for the logistic regression model of Section 4.4.1. For notational convenience, we redefine the binary class labeling convention, adopting a $\{0, 1\}$ labeling scheme, instead of the $\{-1, 1\}$ scheme used in the remainder of this dissertation. Thus, in this section, $y \in \{0, 1\}$ represents a discrete class label, and $z \in \mathbb{R}$ denotes the *score* of a particular linear classification example, $\boldsymbol{x} \in \mathbb{R}^N$, i.e., $z = \langle \boldsymbol{x}, \boldsymbol{w} \rangle$ for some separating hyperplane defined by the normal vector $\boldsymbol{w}$. The logistic likelihood of (4.23) therefore becomes

$$p(y|z) = \frac{\exp(\alpha y z)}{\big(1 + \exp(\alpha z)\big)}. \tag{E.5}$$

In order to compute the sum-product GAMP updates, we must be able to evaluate the posterior mean and variance of a random variable, $z \sim \mathcal{N}(p, \tau_p)$, under the likelihood (E.5). Unfortunately, evaluating the necessary integrals is analytically intractable under the logistic likelihood. Instead, we will approximate the posterior mean, $E[z|y]$, and variance, $\text{var}\{z|y\}$, using a variational approximation technique closely related to that described by Bishop [67, §10.6].

Our goal in variational inference is to iteratively maximize a lower bound on the marginal likelihood

$$p(y) = \int_z p(y|z)p(z)dz. \tag{E.6}$$

In order to do so, we will make use of a variational lower bound on the logistic sigmoid, $\sigma(z) \triangleq \big(1 + \exp(-\alpha z)\big)^{-1}$, namely

$$\sigma(z) \geq \sigma(\xi) \exp\big(\tfrac{\alpha}{2}(z - \xi) - \lambda(\xi)(z^2 - \xi^2)\big), \tag{E.7}$$

where ξ represents the variational parameter that we will optimize in order to maximize the lower bound, and $\lambda(\xi) \triangleq \frac{\alpha}{2\xi}\big(\sigma(\xi) - \frac{1}{2}\big)$. The derivation of this lower bound closely mirrors a similar derivation in [67, §10.5], which considered the case of a fixed sigmoid scaling of unity, i.e., $\alpha = 1$. It is a tedious, but straightforward, bookkeeping exercise to generalize to an arbitrary scale α.

Armed with the variational lower bound of (E.7), we begin by noting that the likelihood of (E.5) can be rewritten as $p(y|z) = e^{\alpha yz}\sigma(-z)$. Applying the variational lower bound, it follows that

$$p(y|z) = e^{\alpha yz}\sigma(-z) \geq e^{\alpha yz}\sigma(\xi)\exp\big(\tfrac{-\alpha}{2}(z+\xi) - \lambda(\xi)(z^2 - \xi^2)\big). \tag{E.8}$$

This leads to the following bound on the joint distribution, $p(y, z)$:

$$p(y, z) = p(y|z)p(z) \geq h(z, \xi)p(z), \tag{E.9}$$

where $h(z, \xi) \triangleq \sigma(\xi)\exp\big(\alpha yz - \frac{\alpha}{2}(z+\xi) - \lambda(\xi)(z^2 - \xi^2)\big)$.

Due to the monotonicity of the logarithm, (E.9) implies that

$$\begin{aligned} \log p(y, z) &\geq \log h(z, \xi) + \log p(z), && \text{(E.10)} \\ &= \alpha yz - \tfrac{\alpha}{2}(z+\xi) - \lambda(\xi)(z^2 - \xi^2) + \log p(z) + \text{const}. && \text{(E.11)} \end{aligned}$$

Replacing $p(z)$ in (E.11) with its Gaussian kernel yields the following bound for the joint distribution:

$$\log p(y, z) \geq \alpha yz - \tfrac{\alpha}{2}(z+\xi) - \lambda(\xi)(z^2 - \xi^2) - \tfrac{1}{2\tau_p}(z-p)^2 + \text{const}. \tag{E.12}$$

From Bayes' rule, we can conclude that

$$\log p(z|y) + \log p(y) \geq \alpha y z - \tfrac{\alpha}{2}(z+\xi) - \lambda(\xi)(z^2 - \xi^2) - \tfrac{1}{2\tau_p}(z-p)^2 + \text{const.} \quad \text{(E.13)}$$

Collecting those terms in (E.13) which are a function of z, it follows that

$$\log p(z|y) \geq -\big(\tfrac{1}{2\tau_p} + \lambda(\xi)\big)z^2 + \big(\tfrac{p}{\tau_p} + \alpha(y - \tfrac{1}{2})\big)z + \text{const.} \quad \text{(E.14)}$$

This quadratic form in z suggests an appropriate variational posterior, $p_v(z|y)$, is a Gaussian, namely:

$$p_v(z|y) = \mathcal{N}(\hat{z}, \tau_z), \quad \text{(E.15)}$$

with

$$\hat{z} \triangleq \tau_z\big(p/\tau_p + \alpha(y - \tfrac{1}{2})\big), \quad \text{(E.16)}$$

$$\tau_z \triangleq \tau_p\big(1 + 2\tau_p\lambda(\xi)\big)^{-1}. \quad \text{(E.17)}$$

Under the variational approximation of the posterior, (E.15), the GAMP sum-product updates become trivial to compute: $E[z|y] \cong \hat{z}$, and $\text{var}\{z|y\} \cong \tau_z$. All that remains is for us to optimize the variational bound of (E.14) by maximizing with respect to the variational parameter, ξ. This can be accomplished efficiently, and in few iterations, via an expectation-maximization (EM) algorithm, as described in [67, §10.6].

At the i^{th} EM iteration, we plug an existing ξ^i into (E.16) and (E.17) in order to compute $\hat{z}^i$ and τ_z^i. Then, letting $\mathcal{Q}^i(\xi) \triangleq E_{Z|Y}\big[\log h(z,\xi)p(z)\big|y;\xi^i\big]$ denote the EM

cost function at the i^{th} iteration, we update ξ as

$$\begin{aligned}
\xi^{i+1} &\triangleq \underset{\xi}{\operatorname{argmax}}\, \mathcal{Q}^i(\xi) && \text{(E.18)}\\
&= \underset{\xi}{\operatorname{argmax}}\, E_{Z|Y}\big[\log h(z,\xi)\big|y;\xi^i\big] && \text{(E.19)}\\
&= \underset{\xi}{\operatorname{argmax}}\, E_{Z|Y}\big[\log \sigma(\xi) - \tfrac{\alpha}{2}\xi - \lambda(\xi)(z^2-\xi^2)\big|y;\xi^i\big] && \text{(E.20)}\\
&= \underset{\xi}{\operatorname{argmax}}\, \log\sigma(\xi) - \tfrac{\alpha}{2}\xi - \lambda(\xi)\big(E_{Z|Y}[z^2|y;\xi^i] - \xi^2\big) && \text{(E.21)}\\
&= \underset{\xi}{\operatorname{argmax}}\, \log\sigma(\xi) - \tfrac{\alpha}{2}\xi - \lambda(\xi)\big(\tau_z^i + |\hat{z}^i|^2 - \xi^2\big). && \text{(E.22)}
\end{aligned}$$

Upon computing the first-order optimality conditions for (E.22) and solving for ξ, we find that

$$\xi^{i+1} = \sqrt{\tau_z^i + |\hat{z}^i|^2}. \tag{E.23}$$

Motivated by (E.23), a reasonable initialization of ξ is given by $\xi^0 = \sqrt{\tau_p + |p|^2}$.

In summary, an accurate and efficient approximation to the sum-product GAMP updates of the logistic regression model can be found by completing a handful of iterative computations of (E.16), (E.17), and (E.23), until convergence is achieved (see the pseudocode of Algorithm 2).

E.3 Sum-Product GAMP Updates for a Hinge Likelihood

In this section, we describe the steps needed to implement the sum-product GAMP message updates for the hinge loss classification model of Section 4.4.3. To begin with, we first observe that $\Theta_H(y,z)$ in (4.26) can be interpreted as the (negative) log-likelihood of the following likelihood distribution:

$$\tilde{p}(y|z) = \frac{1}{\exp(\max(0, 1-yz))}. \tag{E.24}$$

Note that (E.24) is improper because it cannot be normalized to integrate to unity. Nevertheless, we shall see that in the GAMP framework, this is not a problem.

Prior to computing $\mathrm{E}[z|y]$ and $\mathrm{var}\{z|y\}$, we first express the posterior distribution, $p(z|y)$, as

$$p(z|y) = \frac{1}{C_y}\tilde{p}(y|z)p(z), \tag{E.25}$$

where constant C_y is chosen to ensure that $p(z|y)$ integrates to unity. Since y is a binary label, we must consider two cases. First,

$$\begin{aligned}
C_1 &\triangleq \int_z \tilde{p}(z|y=1)p(z)dz \\
&= \int_{-\infty}^{1} \exp(z-1)\mathcal{N}(z;p,\tau_p)dz + \int_1^{\infty} \mathcal{N}(z;p,\tau_p)dz && \text{(E.26)} \\
&= \exp(p+\tfrac{1}{2}\tau_p - 1)\int_{-\infty}^{1} \mathcal{N}(z;p+\tau_p,\tau_p)dz + \int_1^{\infty} \mathcal{N}(z;p,\tau_p)dz && \text{(E.27)} \\
&= \exp(p+\tfrac{1}{2}\tau_p - 1)\Phi\Big(\frac{1-(p+\tau_p)}{\sqrt{\tau_p}}\Big) + \Big[1-\Phi\Big(\frac{1-p}{\sqrt{\tau_p}}\Big)\Big], \\
&= \exp(p+\tfrac{1}{2}\tau_p - 1)\Phi\Big(\frac{1-(p+\tau_p)}{\sqrt{\tau_p}}\Big) + \Phi\Big(\frac{p-1}{\sqrt{\tau_p}}\Big) && \text{(E.28)}
\end{aligned}$$

where $\Phi(\cdot)$ is the CDF of the standard normal distribution. In moving from (E.26) to (E.27), we re-expressed the product of an exponential and Gaussian as the product of a constant and a Gaussian by completing the square. Following the same procedure, we arrive at an expression for C_{-1}:

$$\begin{aligned}
C_{-1} &\triangleq \int_z \tilde{p}(z|y=-1)p(z)dz \\
&= \exp(-p+\tfrac{1}{2}\tau_p - 1)\Phi\Big(\frac{p-\tau_p+1}{\sqrt{\tau_p}}\Big) + \Phi\Big(\frac{-p-1}{\sqrt{\tau_p}}\Big) && \text{(E.29)}
\end{aligned}$$

Now that we have computed the normalizing constant, we turn to evaluating the posterior mean. First considering the $y=1$ case, the posterior mean can be evaluated

as:

$$\begin{aligned}
\mathrm{E}[z|y=1] &= \int_z zp(z|y=1)dz \\
&= \frac{1}{C_1}\int_z z\,\tilde{p}(y=1|z)p(z)dz \\
&= \frac{1}{C_1}\left[\int_{-\infty}^{1} z\exp(z-1)\mathcal{N}(z;p,\tau_p)dz + \int_1^{\infty} z\mathcal{N}(z;p,\tau_p)dz\right] \\
&= \frac{1}{C_1}\left[\exp(p+\tfrac{1}{2}\tau_p-1)\int_{-\infty}^{1} z\mathcal{N}(z;p+\tau_p,\tau_p)dz \right. \\
&\qquad \left. + \int_1^{\infty} z\mathcal{N}(z;p,\tau_p)dz\right] \\
&= \frac{1}{C_1}\left[\exp(p+\tfrac{1}{2}\tau_p-1)\Phi\Big(\frac{1-p-\tau_p}{\sqrt{\tau_p}}\Big)\int_{-\infty}^{1} z\frac{\mathcal{N}(z;p+\tau_p,\tau_p)}{\Phi\Big(\frac{1-p-\tau_p}{\sqrt{\tau_p}}\Big)}dz \right. \\
&\qquad \left. + \Big(1-\Phi\Big(\frac{1-p}{\sqrt{\tau_p}}\Big)\Big)\int_1^{\infty} z\frac{\mathcal{N}(z;p,\tau_p)}{\Big(1-\Phi\Big(\frac{1-p}{\sqrt{\tau_p}}\Big)\Big)}dz\right]. \qquad \text{(E.30)}
\end{aligned}$$

Examining the integrals in (E.30), we see that each integral represents the first moment of a truncated normal random variable, a quantity which can be computed in closed form. Denote by $\mathcal{TN}(z;\mu,\sigma^2,a,b)$ the truncated normal distribution with (non-truncated) mean μ, variance σ^2, and support (a,b). Then, with a slight abuse of notation, (E.30) can be re-expressed as

$$\begin{aligned}
\mathrm{E}[z|y=1] &= \frac{1}{C_1}\left[\exp(p+\tfrac{1}{2}\tau_p-1)\Phi(\alpha_1)\mathrm{E}[\mathcal{TN}(z;p+\tau_p,\tau_p,-\infty,1)]\right. \\
&\qquad \left. +\Phi(-\beta_1)\mathrm{E}[\mathcal{TN}(z;p,\tau_p,1,\infty)]\right], \qquad \text{(E.31)}
\end{aligned}$$

where $\alpha_1 \triangleq \frac{1-p-\tau_p}{\sqrt{\tau_p}}$, and $\beta_1 \triangleq \frac{1-p}{\sqrt{\tau_p}}$. The closed-form expressions for the first moments of the relevant truncated normal random variables are given by [129]

$$\mathrm{E}[\mathcal{TN}(z;p+\tau_p,\tau_p,-\infty,1)] = p+\tau_p-\sqrt{\tau_p}\frac{\phi(\alpha_1)}{\Phi(\alpha_1)}, \qquad \text{(E.32)}$$

$$\begin{aligned}\mathrm{E}[\mathcal{TN}(z;p,\tau_p,1,\infty)] &= p+\sqrt{\tau_p}\frac{\phi(\beta_1)}{1-\Phi(\beta_1)},\\ &= p+\sqrt{\tau_p}\frac{\phi(-\beta_1)}{\Phi(-\beta_1)}\end{aligned} \tag{E.33}$$

where $\phi(\cdot)$ is the standard normal pdf. In a similar fashion, the posterior mean in the $y=-1$ case is given by:

$$\begin{aligned}\mathrm{E}[z|y=-1] &= \frac{1}{C_{-1}}\left[\exp(-p+\tfrac{1}{2}\tau_p-1)\Phi(-\alpha_{-1})\mathrm{E}[\mathcal{TN}(z;p-\tau_p,\tau_p,-1,\infty)]\right.\\ &\qquad \left.+\Phi(\beta_{-1})\mathrm{E}[\mathcal{TN}(z;p,\tau_p,-\infty,-1)]\right],\end{aligned} \tag{E.34}$$

where $\alpha_{-1}\triangleq\frac{-1-p+\tau_p}{\sqrt{\tau_p}}$, and $\beta_{-1}\triangleq\frac{-1-p}{\sqrt{\tau_p}}$. The relevant truncated normal means are

$$\mathrm{E}[\mathcal{TN}(z;p-\tau_p,\tau_p,-1,\infty)] = p-\tau_p+\sqrt{\tau_p}\frac{\phi(-\alpha_{-1})}{\Phi(-\alpha_{-1})}, \tag{E.35}$$

$$\mathrm{E}[\mathcal{TN}(z;p,\tau_p,-\infty,-1)] = p-\sqrt{\tau_p}\frac{\phi(\beta_{-1})}{\Phi(\beta_{-1})}. \tag{E.36}$$

Next, to compute $\mathrm{var}\{z|y\}$, we take advantage of the relationship $\mathrm{var}\{z|y\}=\mathrm{E}[z^2|y]-\mathrm{E}[z|y]^2$ and opt to evaluate $\mathrm{E}[z^2|y]$. Following the same line of reasoning by which we arrived at (E.30), we find

$$\begin{aligned}\mathrm{E}[z^2|y=1] &= \frac{1}{C_1}\left[\exp(p+\tfrac{1}{2}\tau_p-1)\Phi(\alpha_1)\int_{-\infty}^{1}z^2\frac{\mathcal{N}(z;p+\tau_p,\tau_p)}{\Phi(\alpha_1)}dz\right.\\ &\qquad \left.+\Phi(-\beta_1)\int_{1}^{\infty}z^2\frac{\mathcal{N}(z;p,\tau_p)}{\Phi(-\beta_1)}dz\right].\end{aligned} \tag{E.37}$$

Again, we recognize each integral in (E.37) as the second moment of a truncated normal random variable, which can be computed in closed-form from knowledge of the random variable's mean and variance. From the preceding discussion, we have

expressions for the truncated means. The corresponding variances are given by [129]

$$\mathrm{var}\{\mathcal{TN}(z; p+\tau_p, \tau_p, -\infty, 1)\} \;=\; \tau_p\left[1 - \frac{\phi(\alpha_1)}{\Phi(\alpha_1)}\left(\frac{\phi(\alpha_1)}{\Phi(\alpha_1)} + \alpha_1\right)\right], \tag{E.38}$$

$$\mathrm{var}\{\mathcal{TN}(z; p, \tau_p, 1, \infty)\} \;=\; \tau_p\left[1 - \frac{\phi(-\beta_1)}{\Phi(-\beta_1)}\left(\frac{\phi(-\beta_1)}{\Phi(-\beta_1)} - \beta_1\right)\right]. \tag{E.39}$$

Now we need simply replace each integral in (E.37) with the appropriate closed-form computations involving the truncated normal mean and variance, e.g., the first integral would be substituted with $\mathrm{var}\{\mathcal{TN}(z; p+\tau_p, \tau_p, -\infty, 1)\} + \mathrm{E}[\mathcal{TN}(z; p+\tau_p, \tau_p, -\infty, 1)]^2$. Likewise,

$$\begin{aligned}\mathrm{E}[z^2|y=-1] \;=\; & \frac{1}{C_{-1}}\left[\exp(-p+\tfrac{1}{2}\tau_p - 1)\Phi(-\alpha_{-1})\int_{-1}^{\infty} z^2 \frac{\mathcal{N}(z; p-\tau_p, \tau_p)}{\Phi(-\alpha_{-1})}dz\right. \\ & \left. +\Phi(\beta_{-1})\int_{-\infty}^{1} z^2 \frac{\mathcal{N}(z; p, \tau_p)}{\Phi(\beta_1)}dz\right],\end{aligned} \tag{E.40}$$

with

$$\mathrm{var}\{\mathcal{TN}(z; p-\tau_p, \tau_p, -1, \infty)\} = \tau_p\left[1 - \frac{\phi(-\alpha_{-1})}{\Phi(-\alpha_{-1})}\left(\frac{\phi(-\alpha_{-1})}{\Phi(-\alpha_{-1})} - \alpha_{-1}\right)\right], \tag{E.41}$$

$$\mathrm{var}\{\mathcal{TN}(z; p, \tau_p, -\infty, 1)\} \;=\; \tau_p\left[1 - \frac{\phi(\beta_{-1})}{\Phi(\beta_{-1})}\left(\frac{\phi(\beta_{-1})}{\Phi(\beta_{-1})} + \beta_{-1}\right)\right]. \tag{E.42}$$

Finally, for the purpose of adaptive step-sizing within GAMP, we must be able to compute $\mathrm{E}_{z|y}[\log p(y|z)]$ when $z|y \sim \mathcal{N}(\hat{z}, \tau_z)$. Note that it is sufficient for our purposes to know this expectation up to a $\hat{z}$- and τ_z-independent additive constant; we use the relation $\cong$ to indicate equality up to such a constant. Proceeding from the definition of expectation, in the $y = 1$ case:

$$\begin{aligned}\mathrm{E}_{z|y}[\log p(y=1|z)] \;&\triangleq\; \int_z \log p(y=1|z)\mathcal{N}(z; \hat{z}, \tau_z)dz \\ &\cong\; \int_z \log \tilde{p}(y=1|z)\mathcal{N}(z; \hat{z}, \tau_z)dz\end{aligned}$$

$$
\begin{aligned}
&= -\int_z \max(0, 1-z)\mathcal{N}(z;\hat{z},\tau_z)dz \\
&= -\int_{-\infty}^{1}(1-z)\mathcal{N}(z;\hat{z},\tau_z)dz \\
&= -\int_{-\infty}^{1}\mathcal{N}(z;\hat{z},\tau_z)dz + \int_{-\infty}^{1} z\mathcal{N}(z;\hat{z},\tau_z)dz \\
&= \Phi\Big(\frac{1-\hat{z}}{\sqrt{\tau_z}}\Big)\big(-1+\mathrm{E}[\mathcal{TN}(z;\hat{z},\tau_z,-\infty,1)]\big). \qquad \text{(E.43)}
\end{aligned}
$$

A similar derivation in the $y=-1$ case yields

$$
\mathrm{E}_{z|y}[\log p(y=-1|z)] \;\cong\; -\Phi\Big(\frac{1+\hat{z}}{\sqrt{\tau_z}}\Big)\big(1+\mathrm{E}[\mathcal{TN}(z;\hat{z},\tau_z,-1,\infty)]\big), \qquad \text{(E.44)}
$$

completing the sum-product GAMP updates for a hinge loss model.

E.4 Sum-Product GAMP Updates for a Robust-p^* Likelihood

In this section, we derive a method for computing the sum-product GAMP updates for a robust activation function of the form (4.29). Our goal throughout this derivation is to make use of quantities that are already available as part of the standard max-sum GAMP updates for the non-robust likelihood, $p^*(y|z)$. Similar to Section E.3, we must compute the quantities C_y, $\mathrm{E}[z|y]$, and $\mathrm{var}\{z|y\}$ under the robust likelihood (4.29), and a prior $p(z)=\mathcal{N}(p,\tau_p)$.

Beginning with the definition of C_y,

$$
\begin{aligned}
C_y &\triangleq \int p(y|z)p(z)dz \\
&= \gamma + (1-2\gamma)\int p^*(y|z)p(z)dz \\
&= \gamma + (1-2\gamma)C_y^*. \qquad \text{(E.45)}
\end{aligned}
$$

The posterior mean is then found by evaluating

$$\begin{aligned}
\mathrm{E}[z|y] &= \frac{1}{C_y}\int z p(y|z)p(z)dz \\
&= \frac{\gamma}{C_y}\int z p(z)dz + \frac{1-2\gamma}{C_y}\int z p^*(y|z)p(z)dz \\
&= \frac{\gamma}{C_y}p + \frac{1-2\gamma}{C_y}C_y^*\,\mathrm{E}^*[z|y]. \qquad \text{(E.46)}
\end{aligned}$$

Lastly, knowledge of $\mathrm{E}[z^2|y]$ is sufficient for computing $\mathrm{var}\{z|y\}$. It is easily verified that

$$\begin{aligned}
\mathrm{E}[z^2|y] &= \frac{1}{C_y}\int z^2 p(y|z)p(z)dz \\
&= \frac{\gamma}{C_y}\int z^2 p(z)dz + \frac{1-2\gamma}{C_y}\int z^2 p^*(y|z)p(z)dz \\
&= \frac{\gamma}{C_y}(\tau_p + p^2) + \frac{1-2\gamma}{C_y}C_y^*\left(\mathrm{var}^*\{z|y\} + \mathrm{E}^*[z|y]^2\right). \qquad \text{(E.47)}
\end{aligned}$$

E.5 EM Learning of Robust-p^* Label Corruption Probability

In this section, we describe an EM learning procedure to adaptively tune the label corruption probability, γ, of the Robust-p^* likelihood of (4.29) based on available training data, $\{y_m\}_{m=1}^M$. Recall from Section 4.5 we introduced hidden indicator variables, $\{\beta_m\}$, that assume the value 1 if y_m was correctly labeled, and 0 otherwise. Since label corruption occurs according to a Bernoulli distribution with success probability γ, it follows that $p(\boldsymbol{\beta}) = \prod_{m=1}^M \gamma^{1-\beta_m}(1-\gamma)^{\beta_m}$. In addition, the likelihood of label y_m, given z_m and β_m, can be written as $p(y_m|z_m,\beta_m) = p^*_{y_m|z_m}(y_m|z_m)^{\beta_m}\; p^*_{y_m|z_m}(-y_m|z_m)^{1-\beta_m}$.

The EM update at the k^{th} iteration proceeds as follows:

$$\gamma^{k+1} = \underset{\gamma}{\mathrm{argmax}}\,\mathrm{E}_{\boldsymbol{z},\boldsymbol{\beta}|\boldsymbol{y}}\big[\log p(\boldsymbol{y},\boldsymbol{z},\boldsymbol{\beta};\gamma)\big|\boldsymbol{y};\gamma^k\big],$$

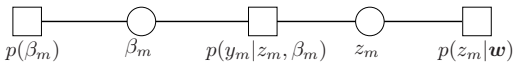


Figure E.1: A factor graph representation of the Robust-p^* hidden data, with circles denoting unobserved random variables, and rectangles denoting pdf "factors".

$$
\begin{aligned}
&= \operatorname*{argmax}_{\gamma} \sum_{m=1}^{M} \mathrm{E}_{z_m,\beta_m|\boldsymbol{y}}\big[\log p(y_m|z_m,\beta_m;\gamma)p(\beta_m;\gamma)\big|\boldsymbol{y};\gamma^k\big], && \text{(E.48)}\\
&= \operatorname*{argmax}_{\gamma} \sum_{m=1}^{M} \mathrm{E}_{z_m,\beta_m|\boldsymbol{y}}\big[\log \gamma^{1-\beta_m}(1-\gamma)^{\beta_m}\big|\boldsymbol{y};\gamma^k\big], && \text{(E.49)}\\
&= \operatorname*{argmax}_{\gamma} \sum_{m=1}^{M} \log(\gamma)\mathrm{E}_{\beta_m|\boldsymbol{y}}\big[1-\beta_m\big|\boldsymbol{y};\gamma^k\big] + \log(1-\gamma)\mathrm{E}_{\beta_m|\boldsymbol{y}}\big[\beta_m\big|\boldsymbol{y};\gamma^k\big],\\
&= \operatorname*{argmax}_{\gamma} \sum_{m=1}^{M} \log(\gamma)\big(1-p(\beta_m=1|\boldsymbol{y};\gamma^k)\big) + \log(1-\gamma)p(\beta_m=1|\boldsymbol{y};\gamma^k) && \text{(E.50)}
\end{aligned}
$$

where in going from (E.48) to (E.49) we kept only those terms that are a function of γ. Differentiating (E.50) w.r.t. γ and setting equal to zero results in the following expression for γ^{k+1}:

$$\gamma^{k+1} = 1 - \frac{1}{M}\sum_{m=1}^{M} p(\beta_m = 1|\boldsymbol{y}). \tag{E.51}$$

In order to compute $p(\beta_m = 1|\boldsymbol{y};\gamma^k)$, we may take advantage of GAMP's factor graph representation. Incorporating the hidden variable β_m into the factor graph, and making z_m explicit, results in the factor graph of Fig. E.1. Let $\nu_{a\to b}(\cdot)$ denote a sum-product message (i.e., a distribution) moving from node a to a connected node b. Then, $\nu_{p(\beta_m)\to\beta_m}(\beta_m) = (\gamma^k)^{1-\beta_m}(1-\gamma^k)^{\beta_m}$, and $\nu_{z_m\to p(y_m|z_m,\beta_m)}(z_m) = \mathcal{N}(z_m;p_m,\tau_m^p)$, which is provided by GAMP. Obeying the rules of the sum-product algorithm, we have that

$$p(\beta_m|\boldsymbol{y};\gamma^k) \quad\propto\quad \nu_{p(\beta_m)\to\beta_m}(\beta_m)\nu_{p(y_m|z_m,\beta_m)\to\beta_m}(\beta_m),$$

$$= \nu_{p(\beta_m)\to\beta_m}(\beta_m)\int_{z_m} p(y_m|z_m,\beta_m)\nu_{z_m\to p(y_m|z_m,\beta_m)}(z_m)dz_m. \quad \text{(E.52)}$$

Evaluating (E.52) for $\beta_m = 1$ gives

$$p(\beta_m = 1|\boldsymbol{y};\gamma^k) \propto (1-\gamma^k)\underbrace{\int_{z_m} p^*_{y|z}(y_m|z_m)\mathcal{N}(z_m;p_m,\tau^p_m)dz_m}_{=C^*_{ym}}, \quad \text{(E.53)}$$

where $C^*_{y_m}$ is a quantity that is computed as a part of the non-robust sum-product GAMP updates for activation function $p^*_{y|z}(y|z)$. Likewise,

$$p(\beta_m = 0|\boldsymbol{y};\gamma^k) \propto \gamma^k(1-C^*_{y_m}) \quad \text{(E.54)}$$

With (E.53) and (E.54), everything necessary to evaluate (E.51) is available.

E.6 EM Update of Logistic Scale, α

Recall from (E.5) that the logistic activation function is parameterized by the scalar α, which controls the steepness of the logistic sigmoid. In this section, we describe an approximate EM procedure for automatically tuning α based on the available training data. Note that in what follows, adopting the convention of Section E.2, $y \in \{0,1\}$. In the course of deriving our EM update procedure, we will make use of $\{z_m\}_{m=1}^M$ as the "hidden data". In addition, we will make use of GAMP's estimate of the posterior mean and variance of z_m, i.e., $\hat{z}_m \triangleq \mathrm{E}_{\mathsf{z}|\mathsf{y}}[\mathsf{z}|\mathsf{y}]$ and $\tilde{z}_m \triangleq \mathrm{var}_{\mathsf{z}|\mathsf{y}}\{\mathsf{z}|\mathsf{y}\}$.

At the k^{th} EM iteration, α^{k+1} is given as the solution to the following optimization problem:

$$\alpha^{k+1} = \underset{\alpha}{\mathrm{argmax}}\, \mathrm{E}_{\mathsf{z}|\mathsf{y}}\big[\log p(\boldsymbol{y},\boldsymbol{z};\alpha)\big|\boldsymbol{y};\alpha^k\big] \quad \text{(E.55)}$$

$$= \operatorname*{argmax}_{\alpha} \sum_{m=1}^{M} \mathrm{E}_{\mathsf{z}_m|\mathsf{y}}\big[\log p(y_m|z_m;\alpha)\big|\boldsymbol{y};\alpha^k\big] \tag{E.56}$$

$$= \operatorname*{argmax}_{\alpha} \sum_{m=1}^{M} \mathrm{E}_{\mathsf{z}_m|\mathsf{y}}\big[\log\big(e^{\alpha y_m z_m}\sigma(-z_m)\big)\big|\boldsymbol{y};\alpha^k\big], \tag{E.57}$$

where $\sigma(z) \triangleq (1+\exp(-\alpha z))^{-1}$. Unfortunately, optimizing (E.57) further is intractable. Instead, we resort to the variational lower-bound approximation of $p(y|z)$ introduced in (E.8), i.e.,

$$\begin{aligned}
\alpha^{k+1} &= \operatorname*{argmax}_{\alpha} \sum_{m=1}^{M} \mathrm{E}_{\mathsf{z}_m|\mathsf{y}}\big[\log\big(e^{\alpha y_m z_m}\sigma(\xi_m)\exp(-\tfrac{\alpha}{2}(z_m+\xi_m)-\lambda(\xi_m)(z_m^2-\xi_m^2))\big)\big|\boldsymbol{y};\alpha^k\big] \\
&= \operatorname*{argmax}_{\alpha} \sum_{m=1}^{M} \alpha y_m \mathrm{E}_{\mathsf{z}_m|\mathsf{y}}[z_m|\boldsymbol{y};\alpha^k] + \log\sigma(\xi_m) - \tfrac{\alpha}{2}\big(\mathrm{E}_{\mathsf{z}_m|\mathsf{y}}[z_m|\boldsymbol{y};\alpha^k]+\xi_m\big) - \\
&\qquad \lambda(\xi_m)\big(\mathrm{E}_{\mathsf{z}_m|\mathsf{y}}[z_m^2|\boldsymbol{y};\alpha^k]-\xi_m^2\big) \qquad\qquad (\text{E.58}) \\
&= \operatorname*{argmax}_{\alpha} \sum_{m=1}^{M} \underbrace{\alpha y_m \hat{z}_m + \log\sigma(\xi_m) - \tfrac{\alpha}{2}(\hat{z}_m+\xi_m) - \lambda(\xi_m)(\tilde{z}_m+\hat{z}_m^2-\xi_m^2)}_{\triangleq Q'_m(\alpha;\xi_m)}. \qquad (\text{E.59})
\end{aligned}$$

Note that the objective function in (E.59) is a function of the variational parameter ξ_m. Using (E.23) as a guide, we specify ξ_m as $\xi_m = \sqrt{\tilde{z}_m+\hat{z}_m^2}$. Making the substitution in (E.59) results in a simplified expression for $Q'_m(\alpha;\xi_m)$:

$$Q'_m(\alpha;\sqrt{\tilde{z}_m+\hat{z}_m^2}) = \alpha y_m \hat{z}_m + \log\sigma\big(\sqrt{\tilde{z}_m+\hat{z}_m^2}\big) - \tfrac{\alpha}{2}\big(\hat{z}_m+\sqrt{\tilde{z}_m+\hat{z}_m^2}\big). \tag{E.60}$$

Furthermore, the derivative w.r.t. α of $Q'_m(\alpha;\xi_m)$ is

$$\frac{\partial}{\partial\alpha}Q'_m(\alpha;\sqrt{\tilde{z}_m+\hat{z}_m^2}) = y_m\hat{z}_m + \frac{\sqrt{\tilde{z}_m+\hat{z}_m^2}}{1+e^{\alpha\sqrt{\tilde{z}_m+\hat{z}_m^2}}} - \tfrac{1}{2}(\hat{z}_m+\sqrt{\tilde{z}_m+\hat{z}_m^2}). \tag{E.61}$$

While a closed-form update of α as the solution of (E.59) is not readily available,

(E.60) and (E.61) can be used in a simple gradient descent optimization strategy to numerically solve for α^{k+1}.

E.7 Bethe Free Entropy-based Update of Logistic Scale, α

Recall from Section 4.5.2 that sum-product GAMP has an interpretation as a Bethe free entropy minimization algorithm. When learning output channel model parameters, one can leverage the relationship between the log-likelihood $\ln p(\boldsymbol{y};\boldsymbol{\theta})$ and the Bethe free entropy $J(f_{\mathsf{w}}, f_{\mathsf{z}})$ for convergent $(f_{\mathsf{w}}, f_{\mathsf{z}})$, which suggests the parameter tuning strategy (4.47). For the logistic activation function (E.5), the required integral in (4.47) remains intractable, thus we again resort to a variational approximation thereof.

Proceeding similar to Appendix E.6, we update α as

$$\alpha^{k+1} = \underset{\alpha}{\operatorname{argmax}} \sum_{m=1}^{M} \log \mathrm{E}_{\mathsf{z}_m|\mathsf{y}}\big[\big(e^{\alpha y_m z_m}\sigma(\xi_m)\exp(-\tfrac{\alpha}{2}(z_m+\xi_m)-\lambda(\xi_m)(z_m^2-\xi_m^2))\big)\big|\boldsymbol{y};\alpha^k\big], \tag{E.62}$$

where the expectation is now with respect to $\mathsf{z}_m|\mathsf{y} \sim \mathcal{N}(\hat{p}_m, \tau_{p_m})$. Recognizing the quadratic form in z_m in (E.62), we complete the square to produce a Gaussian kernel that can be integrated in closed-form. Upon completing this bookkeeping exercise, we find that

$$\alpha^{k+1} = \underset{\alpha}{\operatorname{argmax}} \sum_{m=1}^{M} \log\sigma(\xi_m;\alpha) - \tfrac{1}{2}\log(1+2\tau_{p_m}\lambda(\xi_m;\alpha)) + \delta(\alpha)/\beta(\alpha), \tag{E.63}$$

where

$$\begin{aligned}\delta(\alpha) \;\triangleq\; & \alpha^2\tau_{p_m}(1-2y_m)^2 - 4\alpha\big(2\xi_m\lambda(\xi_m;\alpha)\tau_{p_m} + \xi_m + \hat{p}_m(1-2y_m)\big) \\ & +8\lambda(\xi_m;\alpha)\big(\xi_m^2\big(2\lambda(\xi_m;\alpha)\tau_{p_m}+1\big) - \hat{p}_m^2\big),\end{aligned} \tag{E.64}$$

$$\beta(\alpha) \triangleq 8 + 16\lambda(\xi_m;\alpha)\tau_{p_m}, \tag{E.65}$$

and we have made the dependence of σ and λ (see (E.8)) on α explicit. Equation (E.63) is a difficult expression to further optimize, thus one can resort to an iterative maximization scheme to locate α^{k+1}.

E.8 EM Update of Probit Variance, v^2

In Section 4.4.2 we introduced the probit activation function, (4.24), whose hyperparameter v^2 controls the steepness of the probit sigmoid function. EM learning of the probit standard deviation, v, can be accomplished in a manner similar to that of Section E.6. In particular, v^{k+1} is given as the solution of the following optimization:

$$v^{k+1} = \underset{v}{\operatorname{argmax}} \sum_{m=1}^{M} \underbrace{\mathrm{E}_{\mathsf{z}_m|\mathsf{y}}\big[\log p(y_m|z_m)\big|\boldsymbol{y};v^k\big]}_{\triangleq Q_m(v)}. \tag{E.66}$$

Unfortunately, a closed-form expression for v^{k+1} cannot be obtained due to the difficulty in further simplifying (E.66). Instead, differentiating $Q_m(v)$ w.r.t. v results in

$$\frac{\partial}{\partial v}Q_m(v) = \mathrm{E}_{\mathsf{z}_m|\mathsf{y}}\left[\Big(\frac{-y_m z_m}{v^2}\Big)\phi\Big(\frac{-y_m z_m}{v}\Big)\Big/\Phi\Big(\frac{-y_m z_m}{v}\Big)\Big|\boldsymbol{y};v^k\right]. \tag{E.67}$$

The expectation in (E.67) cannot be evaluated in closed-form. Instead, we evaluate the integral numerically, and use a root-finding bisection search to locate the value of v that sets $\sum \frac{\partial}{\partial v}Q_m(v)$ equal to zero.

Appendix F

MISCELLANEOUS GAMP/TURBOGAMP DERIVATIONS

In this appendix, we provide several useful GAMP/turboGAMP derivations that do not appear in other published work.

F.1 EM Learning of Rate Parameter for a Laplacian Prior

In this section, we derive the EM update of a Laplacian rate parameter from an observation vector, $\boldsymbol{y}$. We assume an i.i.d. signal prior of the form $p(\boldsymbol{x}) = \prod_{n=1}^{N} p(x_n)$, with

$$p(x_n) = \mathcal{L}(x_n; \lambda), \tag{F.1}$$

where $\mathcal{L}(x_n; \lambda) \triangleq \frac{\lambda}{2}\exp(-\lambda|x_n|)$ is a Laplacian distribution with rate λ.

The standard EM update of λ is given as the solution to the following optimization problem:

$$\begin{aligned}
\lambda^{k+1} &= \underset{\lambda}{\operatorname{argmax}}\, \mathrm{E}_{\boldsymbol{x}|\boldsymbol{y}}\big[\log p(\boldsymbol{x};\lambda)\big|\boldsymbol{y};\lambda^k\big] \\
&= \underset{\lambda}{\operatorname{argmax}} \sum_{n=1}^{N} \mathrm{E}_{x_n|\boldsymbol{y}}\big[\log p(x_n;\lambda)\big|\boldsymbol{y};\lambda^k\big] \\
&= \underset{\lambda}{\operatorname{argmax}} \underbrace{N\log\lambda - \lambda\sum_{n=1}^{N}\mathrm{E}_{x_n|\boldsymbol{y}}\big[|x_n|\big|\boldsymbol{y};\lambda^k\big]}_{\triangleq F^k(\lambda)}
\end{aligned} \tag{F.2}$$

Setting the derivative of $F^k(\lambda)$ w.r.t. λ in (F.2) equal to zero, and solving for λ, yields the desired EM update expression:

$$\lambda^{k+1} = N\bigg(\sum_{n=1}^{N} \mathrm{E}_{x_n|\boldsymbol{y}}\big[|x_n|\big|\boldsymbol{y};\lambda^k\big]\bigg)^{-1}. \tag{F.3}$$

Inspection of (F.3) reveals that we must be able to evaluate the posterior expectation of $|x_n|$. To do so, we will make use of GAMP's (approximate) marginal posterior, $p(x_n|\boldsymbol{y})$, which is the product of a Gaussian "extrinsic information" message, $\mathcal{N}(x_n;\hat{r}_n;\tau_n^r)$, and the prior, i.e.,

$$p(x_n|\boldsymbol{y};\lambda^k) = \tfrac{1}{C_n^k}\mathcal{N}(x_n;\hat{r}_n;\tau_n^r)\mathcal{L}(x_n;\lambda^k), \tag{F.4}$$

where C_n^k is a normalization constant defined as

$$C_n^k \triangleq \int_{x_n} \mathcal{N}(x_n;\hat{r}_n;\tau_n^r)\mathcal{L}(x_n;\lambda^k)dx_n. \tag{F.5}$$

Splitting apart the integral in (F.23) gives

$$\begin{aligned}\int_{x_n} \mathcal{N}(x_n;\hat{r}_n;\tau_n^r)\mathcal{L}(x_n;\lambda^k)dx_n \;=\; &\underbrace{\int_{-\infty}^{0} \frac{\lambda^k}{2}\mathcal{N}(x_n;\hat{r}_n,\tau_n^r)\exp(\lambda^k x)dx_n}_{\triangleq \underline{C}_n^k} \\ &+ \underbrace{\int_{0}^{\infty} \frac{\lambda^k}{2}\mathcal{N}(x_n;\hat{r}_n,\tau_n^r)\exp(-\lambda^k x)dx_n}_{\triangleq \bar{C}_n^k}.\end{aligned} \tag{F.6}$$

To evaluate $\underline{C}_n^k$ and $\bar{C}_n^k$, we first complete the square to re-express the multiplication of a Gaussian pdf and a Laplacian pdf as the multiplication of a Gaussian pdf

by a constant, that is,

$$\underline{C}_n^k = \frac{\lambda^k}{2}\exp\left(\frac{\underline{\mu}_n^2 - \hat{r}_n^2}{2\tau_n^r}\right)\int_{-\infty}^{0}\mathcal{N}(x_n;\underline{\mu}_n,\tau_n^r)dx_n, \tag{F.7}$$

$$\bar{C}_n^k = \frac{\lambda^k}{2}\exp\left(\frac{\bar{\mu}_n^2 - \hat{r}_n^2}{2\tau_n^r}\right)\int_{0}^{\infty}\mathcal{N}(x_n;\bar{\mu}_n,\tau_n^r)dx_n, \tag{F.8}$$

where $\underline{\mu}_n \triangleq \hat{r}_n + \lambda^k\tau_n^r$ and $\bar{\mu}_n \triangleq \hat{r}_n - \lambda^k\tau_n^r$. Evaluating the integrals in (F.25) and (F.26) gives

$$\underline{C}_n^k = \frac{\lambda^k}{2}\exp\left(\frac{\underline{\mu}_n^2 - \hat{r}_n^2}{2\tau_n^r}\right)\Phi\left(\frac{-\underline{\mu}_n}{\sqrt{\tau_n^r}}\right), \tag{F.9}$$

$$\begin{aligned}\bar{C}_n^k &= \frac{\lambda^k}{2}\exp\left(\frac{\bar{\mu}_n^2 - \hat{r}_n^2}{2\tau_n^r}\right)\left[1-\Phi\left(\frac{-\bar{\mu}_n}{\sqrt{\tau_n^r}}\right)\right]\\ &= \frac{\lambda^k}{2}\exp\left(\frac{\bar{\mu}_n^2 - \hat{r}_n^2}{2\tau_n^r}\right)\Phi\left(\frac{\bar{\mu}_n}{\sqrt{\tau_n^r}}\right).\end{aligned} \tag{F.10}$$

Now that we have computed the normalizing constant, we turn to evaluating the required posterior expectation:

$$\begin{aligned}\mathrm{E}_{x_n|\boldsymbol{y}}\big[|x_n|\big|\boldsymbol{y};\lambda^k\big] &= \int_{x_n}|x_n|\,p(x_n|\boldsymbol{y};\lambda^k)dx_n,\\ &= \frac{1}{C_n^k}\int_{x_n}|x_n|\,\mathcal{N}(x_n;\hat{r}_n,\tau_n^r)\mathcal{L}(x_n;\lambda^k)dx_n\\ &= \frac{1}{\underline{C}_n^k+\bar{C}_n^k}\left(\frac{\lambda^k}{2}\int_0^\infty x_n\,\mathcal{N}(x_n;\hat{r}_n,\tau_n^r)\exp(-\lambda^k x_n)dx_n\right.\\ &\quad\left.-\frac{\lambda^k}{2}\int_{-\infty}^0 x_n\,\mathcal{N}(x_n;\hat{r}_n,\tau_n^r)\exp(\lambda^k x_n)dx_n\right),\\ &= \frac{1}{\underline{C}_n^k+\bar{C}_n^k}\left(\frac{\lambda}{2}\exp\left(\frac{\bar{\mu}_n^2-\hat{r}_n^2}{2\tau_n^r}\right)\int_0^\infty x_n\,\mathcal{N}(x_n;\bar{\mu}_n,\tau_n^r)dx_n\right.\\ &\quad\left.-\frac{\lambda^k}{2}\exp\left(\frac{\underline{\mu}_n^2-\hat{r}_n^2}{2\tau_n^r}\right)\int_{-\infty}^0 x_n\,\mathcal{N}(x_n;\underline{\mu}_n,\tau_n^r)dx_n\right).\end{aligned} \tag{F.11}$$

In order to evaluate the integrals in (F.34), we first multiply each integral by an

appropriate Gaussian CDF, and its reciprocal:

$$\mathrm{E}_{x_n|\boldsymbol{y}}\big[|x_n|\big|\boldsymbol{y};\lambda^k\big] = \frac{1}{\underline{C}_n^k+\bar{C}_n^k}\left(\bar{C}_n^k\int_0^\infty x_n\,\mathcal{N}(x_n;\bar{\mu}_n,\tau_n^r)\Big/\Phi\Big(\frac{\bar{\mu}_n}{\sqrt{\tau_n^r}}\Big)dx_n - \underline{C}_n^k\int_{-\infty}^0 x_n\,\mathcal{N}(x_n;\underline{\mu}_n,\tau_n^r)\Big/\Phi\Big(\frac{\underline{-\mu}_n}{\sqrt{\tau_n^r}}\Big)dx_n\right). \tag{F.12}$$

Observing the integrals in (F.35), we see that each integral represents the first moment of a truncated normal distribution, a quantity which can be computed in closed form. Denote by $\mathcal{TN}(x;\mu,\sigma^2,a,b)$ a truncated normal random variable with (non-truncated) mean μ, variance σ^2, and support (a,b). Then, with an abuse of notation,

$$\bar{\gamma}_n \triangleq \mathrm{E}[\mathcal{TN}(x_n;\bar{\mu}_n,\tau_n^r,0,\infty)] = \bar{\mu}_n + \sqrt{\tau_n^r}\phi\Big(\frac{\bar{\mu}_n}{\sqrt{\tau_n^r}}\Big)\Big/\Phi\Big(\frac{\bar{\mu}_n}{\sqrt{\tau_n^r}}\Big), \tag{F.13}$$

$$\underline{\gamma}_n \triangleq \mathrm{E}[\mathcal{TN}(x_n;\underline{\mu}_n,\tau_n^r,-\infty,0)] = \underline{\mu}_n - \sqrt{\tau_n^r}\phi\Big(\frac{\underline{-\mu}_n}{\sqrt{\tau_n^r}}\Big)\Big/\Phi\Big(\frac{\underline{-\mu}_n}{\sqrt{\tau_n^r}}\Big), \tag{F.14}$$

where $\phi(\cdot)$ is the standard normal pdf [129]. Applying these facts to (F.35) yields our desired posterior mean:

$$\mathrm{E}_{x_n|\boldsymbol{y}}\big[|x_n|\big|\boldsymbol{y};\lambda^k\big] = \frac{1}{\underline{C}_n^k+\bar{C}_n^k}\Big(\bar{C}_n^k\cdot\bar{\gamma}_n - \underline{C}_n^k\cdot\underline{\gamma}_n\Big). \tag{F.15}$$

F.2 Sum-Product GAMP Equations for an Elastic Net Prior

The `ElasticNetEstimIn` class supports penalized regression problems where the penalty function, $f(\boldsymbol{x})$, in the MAP estimation context, is given by

$$f(\boldsymbol{x}) \triangleq \lambda_1\|\boldsymbol{x}\|_1 + \lambda_2\|\boldsymbol{x}\|_2^2. \tag{F.16}$$

In this section, we describe the steps needed to implement the sum-product message updates within the GAMP framework, enabling MMSE estimation. To begin with,

we first observe that $f(\boldsymbol{x})$ in (F.16) can be interpreted as the log-prior of a separable prior distribution, $p(\boldsymbol{x}) = \prod_n p(x_n)$, in which

$$\begin{aligned} p(x_n) &= \tfrac{1}{\alpha}\tilde{p}(x) && \text{(F.17)} \\ &= \tfrac{1}{\alpha}\mathcal{N}(x_n; 0, (2\lambda_2)^{-1})\mathcal{L}(x_n; \lambda_1), && \text{(F.18)} \end{aligned}$$

where $\tilde{p}(x)$ is an unnormalized distribution, α is the associated normalizing constant, and $\mathcal{L}(x_n; \lambda_1) \triangleq \frac{\lambda_1}{2}\exp(-\lambda_1|x_n|)$ is a Laplacian distribution with rate λ_1.

The sum-product GAMP message updates are given as the posterior mean and variance of a random variable, x, under the prior distribution (F.18), given an observed quantity, $\hat{r}$, obtained as

$$\hat{r} = x + n, \tag{F.19}$$

with $n \sim \mathcal{N}(0, \tau^r)$.

Prior to computing $\mathrm{E}[x|\hat{r}]$ and $\mathrm{var}\{x|\hat{r}\}$, we first express the posterior distribution, $p(x|\hat{r})$, as

$$p(x|\hat{r}) = \frac{1}{C}p(\hat{r}|x)\tilde{p}(x), \tag{F.20}$$

where constant C is chosen to ensure that $p(x|\hat{r})$ integrates to unity. Thus,

$$\begin{aligned} C &\triangleq \int_x p(\hat{r}|x)\tilde{p}(x)dx && \text{(F.21)} \\ &= \int_x \mathcal{N}(x; \hat{r}, \tau^r)\mathcal{N}(x; 0, (2\lambda_2)^{-1})\mathcal{L}(x; \lambda_1)dx && \text{(F.22)} \\ &= \mathcal{N}(0; \hat{r}, \tau^r + (2\lambda_2)^{-1}) \int_x \mathcal{N}\Big(x; \underbrace{\frac{\hat{r}}{2\lambda_2\tau^r + 1}}_{\triangleq\mu}, \underbrace{\frac{\tau^r}{2\lambda_2\tau^r + 1}}_{\triangleq\sigma^2}\Big)\mathcal{L}(x; \lambda_1)dx. && \text{(F.23)} \end{aligned}$$

Splitting apart the integral in (F.23) gives

$$\int_x \mathcal{N}(x;\mu,\sigma^2)\mathcal{L}(x;\lambda_1) = \underbrace{\int_{-\infty}^{0} \frac{\lambda_1}{2}\mathcal{N}(x;\mu,\sigma^2)\exp(\lambda_1 x)dx}_{\triangleq \underline{C}} + \underbrace{\int_{0}^{\infty} \frac{\lambda_1}{2}\mathcal{N}(x;\mu,\sigma^2)\exp(-\lambda_1 x)dx}_{\triangleq \bar{C}}. \quad \text{(F.24)}$$

To evaluate $\underline{C}$ and $\bar{C}$, we first complete the square to re-express the multiplication of a Gaussian pdf and a Laplacian pdf as the multiplication of a Gaussian pdf by a constant, that is,

$$\underline{C} = \frac{\lambda_1}{2}\exp\Big(\frac{\underline{\mu}^2-\mu^2}{2\sigma^2}\Big)\int_{-\infty}^{0}\mathcal{N}(x;\underline{\mu},\sigma^2)dx, \quad \text{(F.25)}$$

$$\bar{C} = \frac{\lambda_1}{2}\exp\Big(\frac{\bar{\mu}^2-\mu^2}{2\sigma^2}\Big)\int_{0}^{\infty}\mathcal{N}(x;\bar{\mu},\sigma^2)dx, \quad \text{(F.26)}$$

where $\underline{\mu} \triangleq \mu+\lambda_1\sigma^2$ and $\bar{\mu} \triangleq \mu-\lambda_1\sigma^2$. Evaluating the integrals in (F.25) and (F.26) gives

$$\underline{C} = \frac{\lambda_1}{2}\exp\Big(\frac{\underline{\mu}^2-\mu^2}{2\sigma^2}\Big)\Phi\Big(\frac{-\underline{\mu}}{\sigma}\Big), \quad \text{(F.27)}$$

$$\bar{C} = \frac{\lambda_1}{2}\exp\Big(\frac{\bar{\mu}^2-\mu^2}{2\sigma^2}\Big)\Big[1-\Phi\Big(\frac{-\bar{\mu}}{\sigma}\Big)\Big] \quad \text{(F.28)}$$

$$= \frac{\lambda_1}{2}\exp\Big(\frac{\bar{\mu}^2-\mu^2}{2\sigma^2}\Big)\Phi\Big(\frac{\bar{\mu}}{\sigma}\Big). \quad \text{(F.29)}$$

Therefore, $C = \mathcal{N}(0;\hat{r},\tau^r+(2\lambda_2)^{-1})(\underline{C}+\bar{C})$.

Now that we have computed the normalizing constant, we turn to evaluating the posterior mean:

$$\mathrm{E}[x|\hat{r}] = \frac{1}{C}\int_x x\, p(\hat{r}|x)\tilde{p}(x)dx, \quad \text{(F.30)}$$

$$= \frac{1}{C}\int_x x\,\mathcal{N}(x;\hat{r},\tau^r)\mathcal{N}(x;0,(2\lambda_2)^{-1})\mathcal{L}(x;\lambda_1)dx \tag{F.31}$$

$$= \frac{\mathcal{N}(0;\hat{r},\tau^r+(2\lambda_2)^{-1})}{C}\int_x x\,\mathcal{N}(x;\mu,\sigma^2)\mathcal{L}(x;\lambda_1)dx \tag{F.32}$$

$$= \frac{1}{\underline{C}+\bar{C}}\left(\frac{\lambda_1}{2}\int_{-\infty}^{0} x\,\mathcal{N}(x;\mu,\sigma^2)\exp(\lambda_1 x)dx \right.$$
$$\left. +\frac{\lambda_1}{2}\int_{0}^{\infty} x\,\mathcal{N}(x;\mu,\sigma^2)\exp(-\lambda_1 x)dx\right), \tag{F.33}$$

$$= \frac{1}{\underline{C}+\bar{C}}\left(\frac{\lambda_1}{2}\exp\left(\frac{\underline{\mu}^2-\mu^2}{2\sigma^2}\right)\int_{-\infty}^{0} x\,\mathcal{N}(x;\underline{\mu},\sigma^2)dx \right.$$
$$\left. +\frac{\lambda_1}{2}\exp\left(\frac{\bar{\mu}^2-\mu^2}{2\sigma^2}\right)\int_{0}^{\infty} x\,\mathcal{N}(x;\bar{\mu},\sigma^2)dx\right). \tag{F.34}$$

In order to evaluate the integrals in (F.34), we first multiply each integral by a specific constant, and its reciprocal:

$$\mathrm{E}[x|\hat{r}] = \frac{1}{\underline{C}+\bar{C}}\left(\frac{\lambda_1}{2}\exp\left(\frac{\underline{\mu}^2-\mu^2}{2\sigma^2}\right)\Phi\left(\frac{-\underline{\mu}}{\sigma}\right)\int_{-\infty}^{0} x\,\frac{\mathcal{N}(x;\underline{\mu},\sigma^2)}{\Phi\left(\frac{-\underline{\mu}}{\sigma}\right)}dx \right.$$
$$\left. +\frac{\lambda_1}{2}\exp\left(\frac{\bar{\mu}^2-\mu^2}{2\sigma^2}\right)\Phi\left(\frac{\bar{\mu}}{\sigma}\right)\int_{0}^{\infty} x\,\frac{\mathcal{N}(x;\bar{\mu},\sigma^2)}{\Phi\left(\frac{\bar{\mu}}{\sigma}\right)}dx\right). \tag{F.35}$$

Observing the integrals in (F.35), we see that each integral represents the first moment of a truncated normal distribution, a quantity which can be computed in closed form. Denote by $\mathcal{TN}(x;\mu,\sigma^2,a,b)$ the truncated normal distribution with (non-truncated) mean μ, variance σ^2, and support (a,b). Then, with an abuse of notation,

$$\mathrm{E}[\mathcal{TN}(x;\underline{\mu},\sigma^2,-\infty,0)] = \underline{\mu}-\sigma\frac{\phi\left(\frac{-\underline{\mu}}{\sigma}\right)}{\Phi\left(\frac{-\underline{\mu}}{\sigma}\right)}, \tag{F.36}$$

$$\mathrm{E}[\mathcal{TN}(x;\bar{\mu},\sigma^2,0,\infty)] = \bar{\mu}+\sigma\frac{\phi\left(\frac{\bar{\mu}}{\sigma}\right)}{\Phi\left(\frac{\bar{\mu}}{\sigma}\right)}, \tag{F.37}$$

where $\phi(\cdot)$ is the standard normal pdf [129]. Applying these facts to (F.35) yields

$$\mathrm{E}[x|\hat{r}] = \frac{1}{\underline{C}+\bar{C}}\Big(\underline{C}\cdot\mathrm{E}[\mathcal{TN}(x;\underline{\mu},\sigma^2,-\infty,0)] + \bar{C}\cdot\mathrm{E}[\mathcal{TN}(x;\bar{\mu},\sigma^2,0,\infty)]\Big). \quad \text{(F.38)}$$

Next, to compute $\mathrm{var}\{x|\hat{r}\}$, we take advantage of the relationship $\mathrm{var}\{x|\hat{r}\} = \mathrm{E}[x^2|\hat{r}] - \mathrm{E}[x|\hat{r}]^2$ and opt to evaluate $\mathrm{E}[x^2|\hat{r}]$. Following the same line of reasoning by which we arrived at (F.35), we reach

$$\begin{aligned}\mathrm{E}[x^2|\hat{r}] &= \frac{1}{\underline{C}+\bar{C}}\left(\frac{\lambda_1}{2}\exp\Big(\frac{\underline{\mu}^2-\mu^2}{2\sigma^2}\Big)\Phi\Big(\frac{-\underline{\mu}}{\sigma}\Big)\int_{-\infty}^{0} x^2\,\frac{\mathcal{N}(x;\underline{\mu},\sigma^2)}{\Phi\Big(\frac{-\underline{\mu}}{\sigma}\Big)}dx\right. \\ &\quad \left. + \frac{\lambda_1}{2}\exp\Big(\frac{\bar{\mu}^2-\mu^2}{2\sigma^2}\Big)\Phi\Big(\frac{\bar{\mu}}{\sigma}\Big)\int_{0}^{\infty} x^2\,\frac{\mathcal{N}(x;\bar{\mu},\sigma^2)}{\Phi\Big(\frac{\bar{\mu}}{\sigma}\Big)}dx\right). \quad \text{(F.39)}\end{aligned}$$

Again, we recognize each integral in (F.39) as the second moment of a truncated normal distribution, which can be computed in closed-form from knowledge of the mean and variance of a truncated normal random variable. From the preceding discussion, we have expressions for the means of the necessary truncated normal random variables. The corresponding variances are given by [129]

$$\mathrm{var}\{\mathcal{TN}(x;\underline{\mu},\sigma^2,-\infty,0)\} = \sigma^2\left[1-\frac{\phi\Big(\frac{-\underline{\mu}}{\sigma}\Big)}{\Phi\Big(\frac{-\underline{\mu}}{\sigma}\Big)}\left(\frac{\phi\Big(\frac{-\underline{\mu}}{\sigma}\Big)}{\Phi\Big(\frac{-\underline{\mu}}{\sigma}\Big)}-\frac{\underline{\mu}}{\sigma}\right)\right], \quad \text{(F.40)}$$

$$\mathrm{var}\{\mathcal{TN}(x;\bar{\mu},\sigma^2,0,\infty)\} = \sigma^2\left[1-\frac{\phi\Big(\frac{\bar{\mu}}{\sigma}\Big)}{\Phi\Big(\frac{\bar{\mu}}{\sigma}\Big)}\left(\frac{\phi\Big(\frac{\bar{\mu}}{\sigma}\Big)}{\Phi\Big(\frac{\bar{\mu}}{\sigma}\Big)}+\frac{\bar{\mu}}{\sigma}\right)\right]. \quad \text{(F.41)}$$

Inserting the needed quantities into (F.39) produces

$$\begin{aligned}\mathrm{E}[x^2|\hat{r}] &= \frac{1}{\underline{C}+\bar{C}}\Big[\underline{C}\Big(\mathrm{var}\{\mathcal{TN}(x;\underline{\mu},\sigma^2,-\infty,0)\}+\mathrm{E}[\mathcal{TN}(x;\underline{\mu},\sigma^2,-\infty,0)]^2\Big) \\ &\quad +\bar{C}\Big(\mathrm{var}\{\mathcal{TN}(x;\bar{\mu},\sigma^2,0,\infty)\}+\mathrm{E}[\mathcal{TN}(x;\bar{\mu},\sigma^2,0,\infty)]^2\Big)\Big], \quad \text{(F.42)}\end{aligned}$$

from which it is possible to calculate $\mathrm{var}\{x|\hat{r}\}$.

Finally, for the purpose of adaptive step-sizing, sum-product GAMP must be able to evaluate the (negative) Kullback-Leibler divergence, $-D_{\mathrm{KL}}(p(x|\hat{r}) \parallel p(x))$, which can be computed as follows:

$$
\begin{aligned}
-D_{\mathrm{KL}}(p(x|\hat{r}) \parallel p(x)) &\triangleq \int_x p(x|\hat{r}) \log\left(\frac{p(x)}{p(x|\hat{r})}\right) dx && \text{(F.43)}\\
&\cong \int_x p(x|\hat{r}) \log\left(\frac{\tilde{p}(x)}{\frac{1}{C}p(\hat{r}|x)\tilde{p}(x)}\right) dx && \text{(F.44)}\\
&= \int_x p(x|\hat{r}) \log\left(\frac{C}{p(\hat{r}|x)}\right) dx && \text{(F.45)}\\
&= \log C - \int_x p(x|\hat{r}) \log \mathcal{N}(x;\hat{r},\tau^r) dx && \text{(F.46)}\\
&= \log C + \tfrac{1}{2}\log(2\pi\tau^r) + \int_x \frac{(x-\hat{r})^2}{2\tau^r} p(x|\hat{r}) dx && \text{(F.47)}\\
&= \log C + \tfrac{1}{2}\log(2\pi\tau^r) + \tfrac{1}{2\tau^r}\left[\mathrm{var}\{x|\hat{r}\} + \mathrm{E}[x|\hat{r}]^2\right]\\
&\quad -\tfrac{\hat{r}}{\tau^r}\mathrm{E}[x|\hat{r}] + \tfrac{\hat{r}^2}{2\tau^r}, && \text{(F.48)}
\end{aligned}
$$

where $\cong$ indicates equality up to an additive constant independent of $\hat{r}$ and τ^r.