Binary Linear Classification and Feature Selection via Generalized Approximate Message Passing

Justin Ziniel, *Student Member, IEEE*, Philip Schniter, *Fellow, IEEE*, and Per Sederberg

***Abstract*—For the problem of binary linear classification and feature selection, we propose algorithmic approaches to classifier design based on the generalized approximate message passing (GAMP) algorithm, recently proposed in the context of compressive sensing. We are particularly motivated by problems where the number of features greatly exceeds the number of training examples, but where only a few features suffice for accurate classification. We show that sum-product GAMP can be used to (approximately) minimize the classification error rate and max-sum GAMP can be used to minimize a wide variety of regularized loss functions. Furthermore, we describe an expectation-maximization (EM)-based scheme to learn the associated model parameters online, as an alternative to cross-validation, and we show that GAMP's state-evolution framework can be used to accurately predict the misclassification rate. Finally, we present a detailed numerical study to confirm the accuracy, speed, and flexibility afforded by our GAMP-based approaches to binary linear classification and feature selection.**

***Index Terms*—Belief propagation, classification, feature selection, message passing, one-bit compressed sensing.**

I. Introduction

In this work we consider binary linear classification and feature selection [1]. 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 vector of quantifiable features $\boldsymbol{x} \in \mathbb{R}^N$ from the sign of a linear "score" $\mathsf{z} \triangleq \langle \boldsymbol{x}, \mathbf{w} \rangle$.[1] 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:

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

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 (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

$$p_{\mathsf{y}_m|\mathsf{z}_m}(1|z_m) = \int_{-\infty}^{z_m} p_{\mathsf{e}}(e)\, de = 1 - p_{\mathsf{y}_m|\mathsf{z}_m}(-1|z_m). \tag{2}$$

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 [2], multi-voxel pattern analysis (MVPA) [3]–[5], conjoint analysis [6], and micro-array gene expression [7]. In MVPA, for instance, neuro-scientists attempt to infer which regions in the human brain are responsible for distinguishing between two cognitive states by measuring neural activity via fMRI at $N \sim 10^4$ voxels. Due to the expensive and time-consuming nature of working with human subjects, classifiers are routinely trained using only $M \sim 10^2$ training examples, and thus $N \gg M$.

In the $N \gg M$ regime, the model of (1) coincides with that of *noisy one-bit compressed sensing* (CS) [8], [9]. In that setting, it is typical to write (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 element-wise $\operatorname{sgn}(\cdot)$, yielding

$$\boldsymbol{y} = \operatorname{sgn}(\boldsymbol{X}\mathbf{w} - \mathbf{e}), \tag{3}$$

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.[2] Importantly, in the $N \gg M$ setting, [9] established performance guarantees on the estimation of K-sparse $\mathbf{w}$ from $O(K \log N/K)$ binary measurements of the form (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.

Manuscript received August 22, 2014; revised December 15, 2014; accepted January 29, 2015. Date of publication February 26, 2015; date of current version March 13, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Fauzia Ahmad. This work was supported by NSF grant CCF-1218754, NSF grant CCF-1018368, DARPA/ONR grant N66001-10-1-4090, and an allocation of computing time from the Ohio Supercomputer Center.

Portions of this work were presented at the Workshop on Information Theory and its Applications, San Diego, CA, USA, 2013 and the 2014 Conference on Information Sciences and Systems, Princeton, NJ, USA, 2014. (*Corresponding Author: P. Schniter.*)

J. Ziniel and P. Schniter are with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210, USA (e-mail: schniter@ece.osu.edu).

P. Sederberg is with the Department of Psychology, The Ohio State University, Columbus, OH 43210 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2015.2407311

[1]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.

[2]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.

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 [10]–[17].

In addition to theoretical analyses, the CS literature also offers a number of high-performance algorithms for the inference of $\mathbf{w}$ in (3), e.g., [8], [9], [18]–[21]. Thus, the question arises as to whether these algorithms also show advantages in the domain 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 [22], which extends the AMP algorithm [23], [24] 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 (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 [25]; *(ii)* Their state-evolutions predict fast convergence rates; *(iii)* They are very flexible with regard to data-modeling assumptions (see, e.g., [26]); *(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 [27].

In this work, we develop a GAMP-based approach to binary linear classification and feature selection that makes the following contributions: 1) in Section II, we show that GAMP implements a particular approximation to the error-rate minimizing linear classifier under the assumed model (1); 2) in Section III, we show that GAMP's state evolution framework can be used to characterize the misclassification rate in the large-system limit; 3) in Section IV, 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 V, we present an EM-based scheme to learn the model parameters online, as an alternative to cross-validation. The numerical study presented in Section VI then confirms the efficacy, flexibility, and speed afforded by our GAMP-based approaches to binary classification and feature selection.

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.

II. GAMP for Classification

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.

A. 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 (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 $\{\hat{y}_t\}_{t=M+1}^{M+T}$,

$$\hat{y}_t = \underset{y_t \in \{-1,1\}}{\arg\max}\, p_{\mathsf{y}_t|\mathbf{y}_{1:M}}\big(y_t \,\big|\, \boldsymbol{y}_{1:M}; \boldsymbol{X}\big), \tag{4}$$

with $\boldsymbol{y}_{1:M} \triangleq [y_1, \ldots, y_M]^{\mathsf{T}}$ and $\boldsymbol{X} \triangleq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{M+T}]^{\mathsf{T}}$. 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) can be computed via the marginalization

$$p_{\mathsf{y}_t|\mathbf{y}_{1:M}}\big(y_t \,\big|\, \boldsymbol{y}_{1:M}; \boldsymbol{X}\big) = p_{\mathsf{y}_t, \mathbf{y}_{1:M}}\big(y_t, \boldsymbol{y}_{1:M}; \boldsymbol{X}\big) C_{\mathbf{y}}^{-1} \tag{5}$$

$$= C_{\mathbf{y}}^{-1} \sum_{\boldsymbol{y} \in \mathcal{Y}_t(y_t)} \int p_{\mathbf{y},\mathbf{w}}(\boldsymbol{y}, \boldsymbol{w}; \boldsymbol{X})\, d\boldsymbol{w} \tag{6}$$

with scaling constant $C_{\mathbf{y}} \triangleq p_{\mathbf{y}_{1:M}}\big(\boldsymbol{y}_{1:M}; \boldsymbol{X}\big)$, label vector $\boldsymbol{y} = [y_1, \ldots, y_{M+T}]^{\mathsf{T}}$, and constraint set

$$\mathcal{Y}_t(y) \triangleq \{\tilde{\boldsymbol{y}} \in \{-1,1\}^{M+T} \text{ s.t. } [\tilde{\boldsymbol{y}}]_t = y \text{ and } [\tilde{\boldsymbol{y}}]_m = y_m \; \forall m = 1, \ldots, M\}$$

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 (6) factors as

$$p_{\mathbf{y},\mathbf{w}}(\boldsymbol{y}, \boldsymbol{w}; \boldsymbol{X}) = \prod_{m=1}^{M+T} p_{\mathsf{y}_m|\mathsf{z}_m}(y_m \,|\, \boldsymbol{x}_m^{\mathsf{T}}\boldsymbol{w}) \prod_{n=1}^{N} p_{\mathsf{w}_n}(w_n) \tag{7}$$

due to the model (1) and assuming a separable prior, i.e.,

$$p_{\mathbf{w}}(\boldsymbol{w}) = \prod_{n=1}^{N} p_{\mathsf{w}_n}(w_n). \tag{8}$$

Although the separability assumption can also be relaxed (see, e.g., [26], [28]), we do not consider such extensions in this work.

The factorization (7) is illustrated using the *factor graph* in Fig. 1(a), 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 (6) is computationally intractable due to the high-dimensional summation and integration, the factor graph in Fig. 1(a) suggests the use of loopy belief propagation (LBP) [29], and in particular the *sum-product algorithm* (SPA) [30], 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

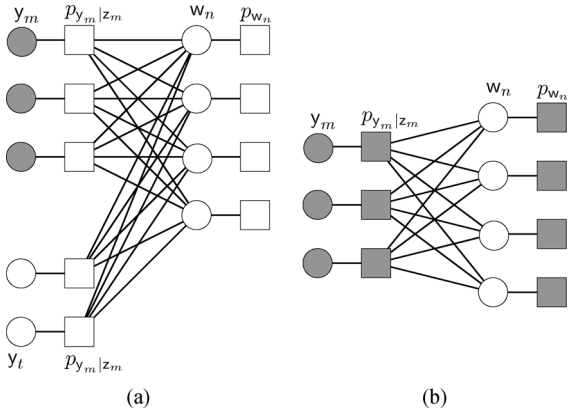

Fig. 1. Factor graph representations of the integrand of (7), with white/grey circles denoting unobserved/observed random variables, and rectangles denoting pdf "factors". (a) Full. (b) Reduced.

loopy graphs, such as turbo decoding [31], computer vision [32], and compressive sensing [22]–[24].

Because a direct application of the SPA to the factor graph in Fig. 1(a) is itself computationally infeasible in the high-dimensional case of interest, we turn to a recently developed approximation: the sum-product variant of GAMP [22], 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 element-wise w.r.t the GAMP-approximated marginal posterior pdfs (with superscript k denoting the iteration)

$$q(z_m \,|\, \hat{p}_m^k, \tau_{p_m}^k) = p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|z_m)\,\mathcal{N}(z_m; \hat{p}_m^k, \tau_{p_m}^k)\,C_{\mathsf{z}}^{-1} \quad (9)$$

$$q(w_n \,|\, \hat{r}_n^k, \tau_{r_n}^k) = p_{\mathsf{w}_n}(w_n)\,\mathcal{N}(w_n; \hat{r}_n^k, \tau_{r_n}^k)\,C_{\mathsf{w}}^{-1} \quad (10)$$

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 element-wise. Due to space limitations, we refer the interested reader to [22] for an overview and derivation of GAMP, to [25] for rigorous analysis under large i.i.d sub-Gaussian $\boldsymbol{X}$, and to [33], [34] for fixed-point and local-convergence analysis under arbitrary $\boldsymbol{X}$.

Applying GAMP to the classification factor graph in Fig. 1(a) 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. 1(b), after which the (approximated) posterior test-label pdfs are computed via

$$p_{\mathsf{y}_t|\boldsymbol{\mathsf{y}}_{1:M}}\left(y_t|\boldsymbol{y}_{1:M}; \boldsymbol{X}\right) \approx \int p_{\mathsf{y}_t|\mathsf{z}_t}(y_t|z_t)\,\mathcal{N}(z_t; \hat{z}_t^\infty, \tau_{z,t}^\infty)\,dz_t \quad (11)$$

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 "$k = \infty$."

B. Max-Sum GAMP

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

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathbb{R}^N}{\arg\min} \sum_{m=1}^{M} f_{\mathsf{z}_m}(\boldsymbol{x}_m^\mathsf{T}\boldsymbol{w}) + \sum_{n=1}^{N} f_{\mathsf{w}_n}(w_n), \quad (12)$$

Algorithm 1: Generalized Approximate Message Passing

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

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

1: **repeat**
2: $\quad\boldsymbol{\tau}_p^k \leftarrow \boldsymbol{S}\boldsymbol{\tau}_w^k$
3: $\quad\hat{\boldsymbol{p}}^k \leftarrow \boldsymbol{X}\hat{\boldsymbol{w}}^k - \hat{\boldsymbol{s}}^{k-1}\boldsymbol{\tau}_p^k$
4: $\quad$**if** `SumProduct` **then**
5: $\quad\quad\hat{\boldsymbol{z}}^k \leftarrow \mathrm{E}\{\mathsf{z}\,|\,\hat{\boldsymbol{p}}^k, \boldsymbol{\tau}_p^k\}$
6: $\quad\quad\boldsymbol{\tau}_z^k \leftarrow \mathrm{var}\{\mathsf{z}\,|\,\hat{\boldsymbol{p}}^k, \boldsymbol{\tau}_p^k\}$
7: $\quad$**else if** `MaxSum` **then**
8: $\quad\quad\hat{\boldsymbol{z}}^k \leftarrow \mathrm{prox}_{\boldsymbol{\tau}_p^k f_{\mathsf{z}_m}}(\hat{\boldsymbol{p}}^k)$
9: $\quad\quad\boldsymbol{\tau}_z^k \leftarrow \boldsymbol{\tau}_p^k \mathrm{prox}'_{\boldsymbol{\tau}_p^k f_{\mathsf{z}_m}}(\hat{\boldsymbol{p}}^k)$
10: $\quad$**end if**
11: $\quad\boldsymbol{\tau}_s^k \leftarrow 1/\boldsymbol{\tau}_p^k - \boldsymbol{\tau}_z^k/(\boldsymbol{\tau}_p^k)^2$
12: $\quad\hat{\boldsymbol{s}}^k \leftarrow (\hat{\boldsymbol{z}}^k - \hat{\boldsymbol{p}}^k)/\boldsymbol{\tau}_p^k$
13: $\quad\boldsymbol{\tau}_r^k \leftarrow 1/(\boldsymbol{S}^\mathsf{T}\boldsymbol{\tau}_s^k)$
14: $\quad\hat{\boldsymbol{r}}^k \leftarrow \hat{\boldsymbol{w}}^k + \boldsymbol{\tau}_r^k \boldsymbol{X}^\mathsf{T}\hat{\boldsymbol{s}}^k$
15: $\quad$**if** `SumProduct` **then**
16: $\quad\quad\hat{\boldsymbol{w}}^{k+1} \leftarrow \mathrm{E}\{\mathsf{w}\,|\,\hat{\boldsymbol{r}}^k, \boldsymbol{\tau}_r^k\}$
17: $\quad\quad\boldsymbol{\tau}_w^{k+1} \leftarrow \mathrm{var}\{\mathsf{w}\,|\,\hat{\boldsymbol{r}}^k, \boldsymbol{\tau}_r^k\}$
18: $\quad$**else if** `MaxSum` **then**
19: $\quad\quad\hat{\boldsymbol{w}}^{k+1} \leftarrow \mathrm{prox}_{\boldsymbol{\tau}_r^k f_{\mathsf{w}_n}}(\hat{\boldsymbol{r}}^k)$
20: $\quad\quad\boldsymbol{\tau}_w^{k+1} \leftarrow \boldsymbol{\tau}_r^k \mathrm{prox}'_{\boldsymbol{\tau}_r^k f_{\mathsf{w}_n}}(\hat{\boldsymbol{r}}^k)$
21: $\quad$**end if**
22: $\quad k \leftarrow k + 1$
23: **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 (12) can be recognized as the *maximum a posteriori* (MAP) estimate of random vector $\mathbf{w}$ given a separable prior $p_{\mathbf{w}}(\cdot)$ and likelihood corresponding to (1), i.e.,

$$p_{\mathbf{y}|\mathbf{w}}(\boldsymbol{y}|\boldsymbol{w}; \boldsymbol{X}) = \prod_{m=1}^{M} p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|\boldsymbol{x}_m^\mathsf{T}\boldsymbol{w}), \quad (13)$$

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. 1(b).

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 [35]. Since max-sum LBP is itself intractable for the high-dimensional problems of interest, we turn to the max-sum variant of GAMP [22], which is also specified in Algorithm 1. There, lines 8–9 are to be interpreted as

$$\hat{z}_m^k = \mathrm{prox}_{\tau_{p_m}^k f_{\mathsf{z}_m}}(\hat{p}_m^k), \;\; m = 1, \ldots, M, \quad (14)$$

$$\tau^k_{z_m} = \tau^k_{p_m} \operatorname{prox}'_{\tau^k_{p_m} f_{z_m}}(\hat{p}^k_m), \quad m = 1, \ldots, M, \tag{15}$$

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

$$\operatorname{prox}_{\tau f}(v) \triangleq \arg\min_{u \in \mathbb{R}} \left[f(u) + \frac{1}{2\tau}(u - v)^2 \right] \tag{16}$$

$$\operatorname{prox}'_{\tau f}(v) = \left(1 + \tau f''(\operatorname{prox}_{\tau f}(v))\right)^{-1}, \tag{17}$$

and lines 19–20 are to be interpreted similarly. It is known [33] that, for *arbitrary* $\boldsymbol{X}$, the fixed points of GAMP correspond to the critical points of the optimization objective (12).

C. 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 $\{p_{\mathsf{y}_t|\boldsymbol{\mathsf{y}}_{1:M}}(y_t|\boldsymbol{y}_{1:M})\}_{t=T+1}^{M+T}$ and finding the MAP weight vector $\hat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} p_{\mathbf{w}|\boldsymbol{\mathsf{y}}_{1:M}}(\boldsymbol{w}|\boldsymbol{y}_{1:M})$, respectively, under the label-generation model (13) [equivalently, (1)] and the separable weight-vector prior (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 IV, we discuss the implementation of these update steps for several popular activation functions, and in Section V, we discuss how the parameters of $p_{\mathsf{y}_m|\mathsf{z}_m}$ and p_{w_n} can be learned online.

III. 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 [22], [25]. 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 II.

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 $(\mathbf{y}, \mathbf{X}, \mathbf{w})$. 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, $p_{\mathbf{w},\hat{\mathbf{w}}^k}(\boldsymbol{w}, \hat{\boldsymbol{w}}^k) \sim \prod_n p_{\mathsf{w}_n,\hat{\mathsf{w}}^k_n}(w_n, \hat{w}^k_n)$ and the first two moments of the joint pdf $p_{\mathsf{w}_n,\hat{\mathsf{w}}^k_n}$ can be computed using [22, Algorithm 3].

Suppose that the $(\mathbf{y}, \mathbf{X})$ 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 $\hat{\mathsf{z}}^k \triangleq \mathbf{x}^{\mathsf{T}}\hat{\mathbf{w}}^k$, respectively. The corresponding test-error rate[3] $\mathcal{E}^k \triangleq \Pr\{\mathsf{y} \neq \operatorname{sgn}(\hat{\mathsf{z}}^k)\}$ 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

$$\begin{aligned} \mathcal{E}^k &= \mathrm{E}\left\{ I_{\{\mathsf{y} \neq \operatorname{sgn}(\hat{\mathsf{z}}^k)\}} \right\} && (18) \\ &= \sum_{y \in \{-1,1\}} \int I_{\{y \neq \operatorname{sgn}(\hat{z}^k)\}} \int p_{\mathsf{y},\hat{\mathsf{z}}^k,\mathsf{z}}(y, \hat{z}^k, z)\, dz\, d\hat{z}^k && (19) \\ &= \sum_{y \in \{-1,1\}} \iint I_{\{y \neq \operatorname{sgn}(\hat{z}^k)\}} p_{\mathsf{y}|\mathsf{z}}(y|z) p_{\mathsf{z},\hat{\mathsf{z}}^k}(z, \hat{z}^k)\, dz\, d\hat{z}^k. && (20) \end{aligned}$$

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

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

where $\xrightarrow{d}$ indicates convergence in distribution. In [36], it is shown that the above matrix components are

$$\Sigma^k_{11} = \delta^{-1}\left(\operatorname{var}\{\mathsf{w}_n\} + \mathrm{E}[\mathsf{w}_n]^2\right), \tag{22}$$

$$\Sigma^k_{12} = \Sigma^k_{21} = \delta^{-1}\left(\operatorname{cov}\{\mathsf{w}_n, \widehat{\mathsf{w}}^k_n\} + \mathrm{E}[\mathsf{w}_n]\mathrm{E}[\widehat{\mathsf{w}}^k_n]\right), \tag{23}$$

$$\Sigma^k_{22} = \delta^{-1}\left(\operatorname{var}\{\widehat{\mathsf{w}}^k_n\} + \mathrm{E}[\widehat{\mathsf{w}}^k_n]^2\right) \tag{24}$$

for label-to-feature ratio δ. As described earlier, the above moments can be computed using [22, Algorithm 3]. The integral in (20) can then be computed (numerically if 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 $\{\hat{y}^{\infty}_t\}_{t=M+1}^{M+T}$ with $\hat{y}^{\infty}_t = \operatorname{sgn}(\boldsymbol{x}_t^{\mathsf{T}}\hat{\boldsymbol{w}}^{\infty})$ were computed and compared to the true test-labels in order to calculate the test-error rate for that realization. Fig. 2(a) 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. 2(b) plots average empirical mean-squared error (MSE) versus state-evolution predicted MSE, where $\mathrm{MSE} = \frac{1}{N}\mathrm{E}\{\|\hat{\boldsymbol{w}}^{\infty} - \boldsymbol{w}\|_2^2\}$.

In both Fig. 2(a) and (b), 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. 2(a) and (b) show very close agreement between empirical averages and state-evolution predictions.

IV. GAMP Nonlinear Steps

Section II 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

[3]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.

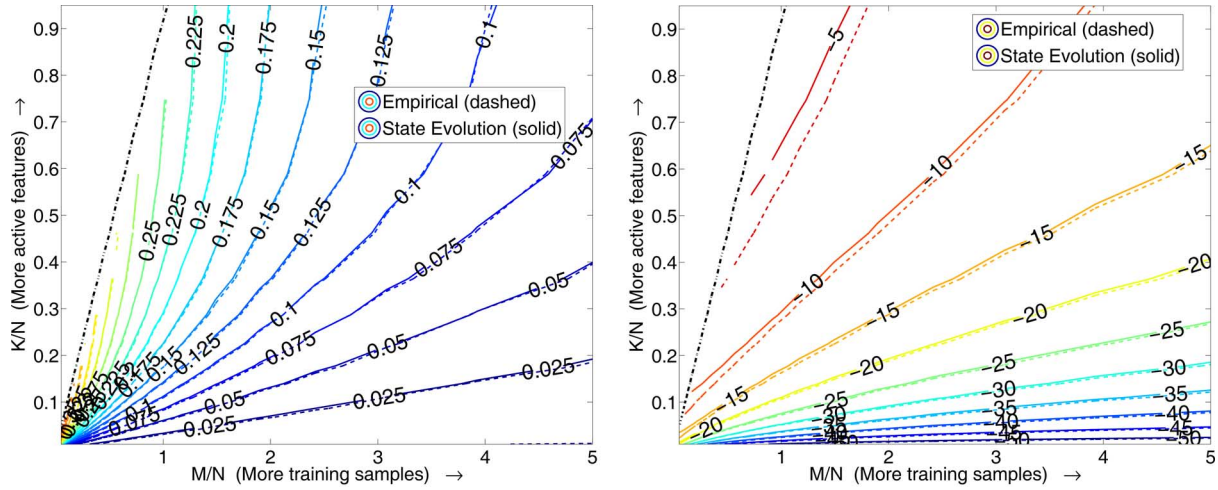

Fig. 2. Test-error rate (a) and weight-vector MSE (b), versus training-to-feature ratio M/N and discriminative-feature ratio K/N, calculated using empirical averaging (dashed) and state-evolution prediction (solid), assuming i.i.d Bernoulli-Gaussian weight vectors and a probit activation function. (a) Test-Error Rate. (b) Weight-Vector MSE (dB).

and 16–17 are computed based on the pdfs in (9) and (10), respectively, and for max-sum GAMP the prox steps in 8–9 are computed using (14)-(15) and those in 19–20 are computed similarly.

A. Logistic Activation Function

Arguably the most popular activation function for binary linear classification is the logistic sigmoid [1, §4.3.2],[37]:

$$p_{\mathsf{y}|\mathsf{z}}(y|z;\alpha) = \frac{1}{1+\exp(-y\alpha z)}, \quad y \in \{-1,1\} \tag{25}$$

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 (9) using the variational approach in Algorithm 2, whose derivation is relegated to [36] for reasons of space. We note that Algorithm 2 is reminiscent of the one presented in [1, §10.6], but is more general in that it handles $\alpha \neq 1$.

For logistic max-sum GAMP, $\hat{z}$ from (14) solves the scalar minimization problem (16) with $f(u) = -\log p_{\mathsf{y}|\mathsf{z}}(y|u;\alpha)$ from (25), 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 (15) can then be computed in closed form using $\hat{z}$ and $f''(\cdot)$ via (17). Note that, unlike the classical ML-based approach to logistic regression (e.g., [1, §4.3.3]), GAMP performs only scalar minimizations and thus does not need to construct or invert a Hessian matrix.

B. Probit Activation Function

Another popular activation function is the probit [1, §4.3.5]:

$$p_{\mathsf{y}|\mathsf{z}}(1|z;v) = \int_{-\infty}^{z} \mathcal{N}(\tau;0,v)d\tau = \Phi\left(\frac{z}{\sqrt{v}}\right) \tag{26}$$

where $p_{\mathsf{y}|\mathsf{z}}(-1|z) = 1 - p_{\mathsf{y}|\mathsf{z}}(1|z) = \Phi\left(-\frac{z}{\sqrt{v}}\right)$ 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 (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 (26). A derivation in [38, §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 I. Max-sum GAMP computation of $(\hat{z}, \tau_z)$ can be performed using a bisection search akin to that described in Section IV-A.

TABLE I
SUM-PRODUCT GAMP COMPUTATIONS FOR PROBIT ACTIVATION FUNCTION

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

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

Input: Class label $y \in \{-1,1\}$, logistic scale α, and GAMP-computed parameters $\hat{p}$ and τ_p (see (9))

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

1: **repeat**

2: $\quad \sigma \leftarrow \left(1+\exp(-\alpha\xi)\right)^{-1}$

3: $\quad \lambda \leftarrow \frac{\alpha}{2\xi}\left(\sigma - \frac{1}{2}\right)$

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

TABLE II
SUM-PRODUCT GAMP COMPUTATIONS FOR THE HINGE-LOSS ACTIVATION FUNCTION. SEE APPENDIX A FOR DEFINITIONS OF γ_y, $\underline{\mu}_y$, $\bar{\mu}_y$, $\underline{v}_y$, $\bar{v}_y$

Quantity	Value
$\hat{z}$	$(1+\gamma_y)^{-1}\underline{\mu}_y + (1+\gamma_y^{-1})^{-1}\bar{\mu}_y$
τ_z	$(1+\gamma_y)^{-1}(\underline{v}_y+\underline{\mu}_y^2)+(1+\gamma_y^{-1})^{-1}(\bar{v}_y+\bar{\mu}_y^2)-\hat{z}^2$

TABLE III
SUM-PRODUCT GAMP COMPUTATIONS FOR A ROBUSTIFIED ACTIVATION FUNCTION. SEE TEXT FOR DEFINITIONS OF C_y^*, $\hat{z}^*$, AND τ_z^*

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

C. Hinge-Loss Activation Function

The *hinge loss* $f_{\mathsf{z}_m}(z) \triangleq \max(0, 1-y_m z)$ is commonly used in the *support vector machine* (SVM) approach to maximum-margin classification [1, §7.1], i.e.,

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_{m=1}^{M} f_{\mathsf{z}_m}(\boldsymbol{x}_m^{\mathsf{T}}\boldsymbol{w}) + \lambda\|\boldsymbol{w}\|_2^2 \tag{27}$$

or variations where $\|\boldsymbol{w}\|_2^2$ is replaced with a sparsity-inducing alternative like $\|\boldsymbol{w}\|_1$ [39]. Recalling Section II-B, this loss leads to the activation function

$$p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|z) \propto \exp\big(-\max(0, 1-y_m z)\big). \tag{28}$$

For hinge-loss sum-product GAMP, the mean and variance $(\hat{z}, \tau_z)$ of (9) can be computed in closed form using the procedure described in Appendix A, and summarized in Table II. Meanwhile, for max-sum GAMP, the proximal steps (14)-(15) can be efficiently computed using bisection search, as in the logistic and probit cases.

D. A Method to Robustify Activation Functions

In some applications, a fraction $\gamma \in (0,1)$ of the training labels are known[4] 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 [40] proposed to use

$$p_{\mathsf{y}|\mathsf{z}}(y|z;\gamma) = (1-\gamma)p^*_{\mathsf{y}|\mathsf{z}}(y|z) + \gamma p^*_{\mathsf{y}|\mathsf{z}}(-y|z) \tag{29}$$

$$= \gamma + (1-2\gamma)p^*_{\mathsf{y}|\mathsf{z}}(y|z). \tag{30}$$

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 (30).

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 III. (See [36] for details.)

In the max-sum case, computing $\hat{z}$ in (14) involves solving the scalar minimization problem in (16) with $f(u) = -\log p_{\mathsf{y}|\mathsf{z}}(y|u;\gamma) = -\log[\gamma + (1-2\gamma)p^*_{\mathsf{y}|\mathsf{z}}(y|u)]$.

[4]A method to learn an unknown γ will be proposed in Section V.

TABLE IV
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 V FOR DEFINITIONS OF $\underline{C}$, $\bar{C}$, $\underline{\mu}$, $\bar{\mu}$, ETC

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

As before, we use a bisection search to find $\hat{z}$ and then we use $f''(\hat{z})$ to compute τ_z via (17).

E. 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 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.,

$$p_{\mathsf{w}_n}(w) = (1-\pi_n)\delta(w) + \pi_n \tilde{p}_{\mathsf{w}_n}(w), \tag{31}$$

where $\delta(\cdot)$ is the Dirac delta function, $\pi_n \in [0,1]$ is the prior[5] 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 [28] and Bernoulli-Gaussian-mixture [27] are common choices, Section VI suggests that Bernoulli-Laplacian also performs well.

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

$$\hat{w} = \mathrm{prox}_{\tau_r f_{\mathsf{w}_n}}(\hat{r}) \tag{32}$$

$$\tau_w = \tau_r\, \mathrm{prox}'_{\tau_r f_{\mathsf{w}_n}}(\hat{r}) \tag{33}$$

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 [23], $f_{\mathsf{w}_n}(w) = \lambda_2 w^2$ for ℓ_2 regularization [22], and $f_{\mathsf{w}_n}(w) = \lambda_1|w| + \lambda_2 w^2$ for the "elastic net" [41]. As described in Section II-B, any regularizer 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 VII, 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 [36]. For the Bernoulli-Laplacian case, these results can be combined with the Bernoulli-$\tilde{p}$ extension in Table VII.

F. The GAMPmatlab Software Suite

The GAMP iterations from Algorithm 1, including the nonlinear steps discussed in this section, have been implemented

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

TABLE V
DEFINITIONS OF ELASTIC-NET QUANTITIES USED IN TABLE IV

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

TABLE VI
ACTIVITY-FUNCTIONS AND THEIR GAMPMATLAB SUM-PRODUCT AND MAX-SUM IMPLEMENTATION METHOD:CF = closed form, VI = variational inference, RF = root-finding

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 VII
WEIGHT-COEFFICIENT PRIORS AND THEIR GAMPMATLAB SUM-PRODUCT AND MAX-SUM IMPLEMENTATION METHOD: CF = closed form, NI = not implemented, NA = not applicable

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

in the open-source "GAMPmatlab" software suite.[6] For convenience, the existing activation-function implementations are summarized in Table VI and relevant weight-prior implementations appear in Table VII.

V. ONLINE PARAMETER TUNING

The activation functions and weight-vector priors described in Section IV depend on modeling parameters that, in practice, must be tuned. For example, the logistic sigmoid (25) 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., [42]).

As an alternative to CV, we consider *online learning* of the unknown model parameters $\boldsymbol{\theta}$ using the methodology from [27], [43]. Here, the goal is to compute the maximum-likelihood estimate $\hat{\boldsymbol{\theta}}_{\mathsf{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\mathbf{y}}(\boldsymbol{y};\boldsymbol{\theta})$, where our data model implies a likelihood function of the form

$$p_{\mathbf{y}}(\boldsymbol{y};\boldsymbol{\theta}) = \int_{\boldsymbol{w}} \prod_m p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|\boldsymbol{x}^{\mathsf{T}}\boldsymbol{w};\boldsymbol{\theta}) \prod_n p_{\mathsf{w}_n}(w_n;\boldsymbol{\theta}). \quad (34)$$

Because it is computationally infeasible to evaluate and/or maximize (34) directly, we apply the expectation-maximization (EM) algorithm [44]. For EM, we treat $\mathbf{w}$ as the "hidden" data, giving the iteration-j EM update

$$\boldsymbol{\theta}^j = \arg\max_{\boldsymbol{\theta}} \mathrm{E}_{\mathbf{w}|\mathbf{y}}\left\{ \log\, p_{\mathbf{y},\mathbf{w}}(\boldsymbol{y},\mathbf{w};\boldsymbol{\theta}) \,\middle|\, \boldsymbol{y};\boldsymbol{\theta}^{j-1}\right\} \quad (35)$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_m \mathrm{E}_{\mathsf{z}_m|\mathbf{y}}\left\{ \log p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|\mathsf{z}_m;\boldsymbol{\theta}) \,\middle|\, \boldsymbol{y};\boldsymbol{\theta}^{j-1}\right\} + \sum_n \mathrm{E}_{\mathsf{w}_n|\mathbf{y}}\left\{ \log p_{\mathsf{w}_n}(\mathsf{w}_n;\boldsymbol{\theta}) \,\middle|\, \boldsymbol{y};\boldsymbol{\theta}^{j-1}\right\}. \quad (36)$$

Furthermore, to evaluate the conditional expectations in (36), GAMP's posterior approximations from (9)-(10) are used. It was shown in [45] 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 [27], [43] 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 (36) 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 work, we propose EM-based learning of the activation-function parameters, i.e., α in the logistic model (25), v in the probit model (26), and γ in the robust model (30). Starting with α, we find that a closed-form expression for the value maximizing (36) remains out of reach, due to the form of the logistic model (25). So, we apply the same variational lower bound used for Algorithm 2, and find that the lower-bound maximizing value of α obeys (see [36])

$$0 = \sum_m \frac{1}{2}(\hat{z}_m y_m - \xi_m) + \frac{\xi_m}{1+\exp(\alpha\xi_m)}, \quad (37)$$

where ξ_m is the variational parameter being used to optimize the lower-bound and $\hat{z}_m \approx \mathrm{E}\{\mathsf{z}_m|\mathbf{y}=\boldsymbol{y}\}$ is output by Algorithm 2. We then solve for α using Newton's method.

To tune the probit parameter, v, we zero the derivative of (36) w.r.t v to obtain

$$0 = \sum_m \mathrm{E}_{\mathsf{z}_m|\mathbf{y}}\left\{ \frac{\partial}{\partial v}\log p_{\mathsf{y}_m|\mathsf{z}_m}(y_m|\mathsf{z}_m;v^j) \,\middle|\, \boldsymbol{y};v^{j-1}\right\} \quad (38)$$

$$= \sum_m \mathrm{E}_{\mathsf{z}_m|\mathbf{y}}\left\{ \frac{-\ddot{c}_m(v^j)}{v^j}\phi(\ddot{c}_m(v^j))\Phi(\ddot{c}_m(v^j))^{-1} \Big| \boldsymbol{y};v^{j-1}\right\}, \quad (39)$$

where $\ddot{c}_m(v) \triangleq (y_m z_m)/v$. We then numerically evaluate the expectation and apply an iterative root-finding procedure to find the EM update v^j that solves (39).

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

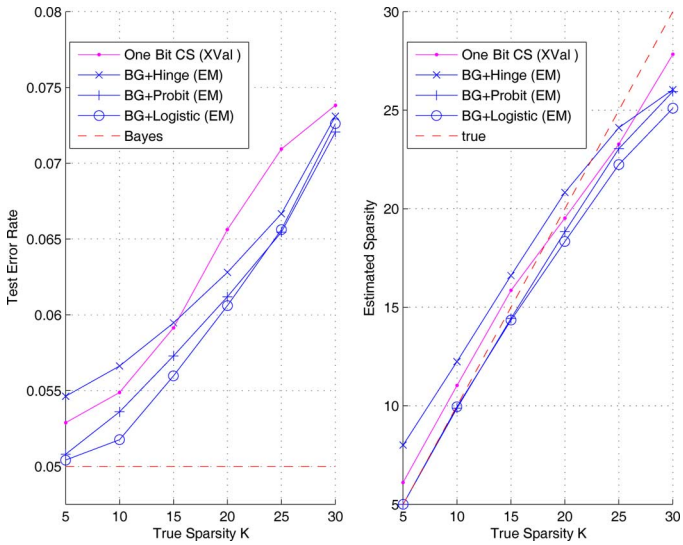

Fig. 3. Test error rate and estimated sparsity $\hat{K}$ for cross-validation-tuned OneBitCS, and for EM-tuned sum-product GAMP classifiers based on the Bernoulli-Gaussian (BG) prior and the hinge ($\times$), probit ($+$), and logistic ($\circ$) activation functions, as a function of the true sparsity K. Here, $N = 30\,000$, $M = 300$, and Bayes error rate was $\varepsilon_{\mathrm{B}} = 0.05$.

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 can be shown [36] that the update of the γ parameter reduces to

$$\gamma^j = \arg\max_{\gamma\in[0,1]} \sum_{m=1}^{M} \mathrm{E}_{\beta_m|\boldsymbol{y}}\big[\log\, p(\beta_m;\gamma)\,\big|\,\boldsymbol{y};\boldsymbol{\theta}^{j-1}\big] \quad (40)$$

$$= \frac{1}{M}\sum_{m=1}^{M} p(\beta_m = 0\,|\,\boldsymbol{y};\boldsymbol{\theta}^{j-1}), \quad (41)$$

where (41) 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.

VI. Numerical Study

In this section we describe several synthetic and real-world 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, 8 MB cache) and 8 GB DDR3 RAM.

A. Synthetic Classification in the $N \gg M$ Regime

We first examine a synthetic problem where the number of features, N, greatly exceeds the number of training examples, M. As discussed in the Introduction, it is possible to perform accurate classification when $N \gg M$ if the number of discriminatory features K is sufficiently small. In this experiment, we consider $N = 30\,000$, $M = 300$, and $K \in \{5,\ldots,30\}$, where the range on K is chosen based on the following information-theoretic argument: M training labels bring $\log_2 M$ bits of information, whereas at least $K\log_2(N/K) \le \log_2\binom{N}{K}$ bits of information are needed to determine the N-length K-sparse Bayes weight vector, assuming that we have no prior knowledge of its support, which takes on $\binom{N}{K}$ possibilities. With $N = 30\,000$ and $M = 300$, it turns out that $K = 31$ is the largest value of $K \le N$ such that $M \ge K\log_2(N/K)$.

Our experiment was of a Monte-Carlo form. In each trial, we constructed a random K-sparse Bayes weight vector $\boldsymbol{w}$ with a support drawn uniformly at random and with non-zero-coefficient amplitudes drawn uniformly in $\{-1,1\}$. We used ± 1 amplitudes to eliminate the potential ambiguity about whether a given non-zero coefficient was effectively non-zero, since, e.g., Gaussian-distributed amplitudes can be arbitrarily close to zero. We then constructed a balanced set of training labels $y_m \in \{-1,1\}$ (i.e., exactly $M/2$ labels were positive) and we drew M i.i.d random feature vectors $\boldsymbol{x}_m$ from the class-conditional generative distribution $\boldsymbol{x}_m|y_m \sim \mathcal{N}(y_m\boldsymbol{w}, v\boldsymbol{I})$.

Fig. 3 shows both the average test error rate and the average estimated sparsity $\hat{K}$ for cross-validation tuned "OneBitCS" from [9],[7] and for EM-tuned sum-product GAMP classifiers based on the Bernoulli-Gaussian (BG) prior and activation functions including hinge loss (HL), probit (PR), and logistic (LR). The average was computed over 1000 Monte-Carlo trials, where in each trial the expected error probability of the designed classifier $\hat{\boldsymbol{w}}$ was computed in closed form as $\Phi(-\boldsymbol{w}^{\mathsf{T}}\hat{\boldsymbol{w}}/\sqrt{v\|\hat{\boldsymbol{w}}\|^2})$. The figure shows all algorithms under test performing relatively close to the Bayes error rate, and for small K it shows BG-LR and BG-PR GAMP performing extremely close to the Bayes error rate. Comparing the classifiers, we see that GAMP's BG-LR performs the best, which is not surprising since the logistic activation function is statistically matched to data model in this experiment [37]. Meanwhile, GAMP's BG-PR classifier performed the second best, and the two remaining classifiers (GAMP's BG-HL and OneBitCS) performed only slightly worse.

Fig. 3 also shows the sparsities estimated by cross-validation in the case of OneBitCS and by the EM-tuning in the case of GAMP. Since the weights returned by sum-product BG-GAMP are non-zero with probability one, the estimated sparsity is defined as the number of coefficients with posterior support probability $p(w_n \neq 0|\boldsymbol{y})$ exceeding 1/2. The figure shows that all algorithms under test returned accurate estimates of the true sparsity K. For small values of K, the estimates returned by BG-LR and BG-PR GAMP were extremely accurate while those for OneBitCS and BG-HL GAMP slightly overestimated the sparsity. Meanwhile, for large values of K, all algorithms underestimated the sparsity by about 15%.

B. Text Classification and Adaptive Learning

We next consider a binary text classification problem based on the Reuter's Corpus Volume I (RCV1) dataset [46]. As in [17], [47], newswire article topic codes CCAT and ECAT were combined to form the positive class while GCAT and MCAT were combined to form constitute the negative class.[8] Although the original dataset consisted of 20 242 balanced training examples of $N = 47\,236$ features, with 677 399 examples reserved for testing, we followed the approach in [17], [47] and swapped training and testing sets in order to test computational efficiency on a large training dataset (and thus $M = 677\,399$).

[7]For cross-validation of OneBitCS, we used 2 folds and searched over all sparsities in a radius of 10 from the true sparsity K.

[8]Data was taken from http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html.

TABLE VIII
A COMPARISON OF DIFFERENT CLASSIFIERS ON THE "SWAPPED" RCV1 BINARY DATASET (WHERE $M \gg N$), SHOWING THE TEST-SET CLASSIFICATION ACCURACY, THE TOTAL AND POST-TUNING RUNTIMES, AND THE DENSITY OF THE WEIGHT VECTOR. BELOW, sp = sum-product; ms = max-sum; BG = Bernoulli-Gaussian; PR = Probit; HL = Hinge loss; L1 = ℓ_1 REGULARIZATION; LR = Logistic

Classifier	Tuning	Accuracy	Runtime (s)	Density
spGAMP: BG-PR	EM	97.4%	**105** / 105	8.6%
spGAMP: BG-HL	EM	97.3%	134 / 134	8.9%
msGAMP: L1-LR	EM	97.6%	684 / 123	9.8%
msGAMP: L1-LR	xval	97.6%	3068 / 278	19.6%
CDN [17]	xval	**97.7%**	1298 / 112	10.9%
TRON [49]	xval	**97.7%**	1682 / 133	10.8%
TFOCS [48]	xval	97.6%	1086 / 94	19.2%
OneBitCS [9]	xval	90.1%	193 / **1**	**1.3%**

TABLE IX
A COMPARISON OF DIFFERENT CLASSIFIERS ON THE "NON-SWAPPED" RCV1 BINARY DATASET (WITH $N > M$), SHOWING THE TEST-SET CLASSIFICATION ACCURACY, THE TOTAL AND POST-TUNING RUNTIMES, AND THE DENSITY OF THE WEIGHT VECTOR. BELOW, sp = sum-product; ms = max-sum; BG = Bernoulli-Gaussian; PR = Probit; HL = Hinge loss; L1 = ℓ_1 REGULARIZATION; LR = Logistic

Classifier	Tuning	Accuracy	Runtime (s)	Density
spGAMP: BG-PR	EM	95.5%	**4** / 4	7.4%
spGAMP: BG-HL	EM	95.1%	6 / 6	3.1%
msGAMP: L1-LR	EM	95.6%	16 / 3	1.8%
msGAMP: L1-LR	xval	95.5%	134 / 16	4.6%
CDN [17]	xval	95.5%	11 / 2	5.0%
TRON [49]	xval	**96.0%**	19 / 3	12.4%
TFOCS [48]	xval	95.7%	17 / 2	4.3%
OneBitCS [9]	xval	89.7%	8 / **0.1**	**0.8%**

As in [17], we constructed feature vectors as cosine-normalized logarithmic transformations of the TF-IDF (term frequency – inverse document frequency) data vectors. We note that the resulting features are very sparse; only 0.16% of the entries in $\boldsymbol{X}$ are non-zero. Finally, we trained linear classifiers (i.e., weight vectors) using four GAMP-based methods and four existing state-of-the-art methods: TFOCS [48] in L1-LR mode, CDN [17], TRON [49], and OneBitCS [9]. In doing so, for EM learning we used 5 EM iterations, and for cross-validation we used 2 folds and a logarithmically spaced grid of size 10.[9]

Table VIII summarizes the performance achieved by the resulting classifiers, including the test-set classification accuracy, weight-vector density (i.e., the fraction of non-zero weights), and two runtimes: the *total* runtime needed to train the classifier, which includes EM- or cross-validation-based parameter tuning, and the *post-tuning* runtime. Although it is customary to report only the latter, we feel that the former better captures the true computational cost of classifier design. We note that, in the case of spGAMP, the total and post-tuning runtime are identical because EM tuning was performed once per GAMP iteration. In contrast, for msGAMP, we ran many GAMP iterations per EM iteration, and hence the total runtime (which avoids EM iterations) is much longer. We also note that the post-tuning runtime of OneBitCS is extremely fast because of a computational trick that we learned via personal communication with an author, Yaniv Plan: Given signed labels $y_m \in \{-1, 1\}$ and a sparsity estimate $\hat{K}$, the OneBitCS weight vector $\hat{\boldsymbol{w}}$ can be computed from the training pair $(\boldsymbol{X}, \boldsymbol{y})$ via $\hat{\boldsymbol{w}} = \mathrm{thresh}_{\hat{K}}(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{y})$, where $\mathrm{thresh}_{\hat{K}}(\cdot)$ is the mapping from $\mathbb{R}^N \to \mathbb{R}^N$ that preserves the input components with the largest $\hat{K}$ magnitudes and zeros the remainder.

Table VIII shows all 8 classifiers achieving nearly identical test-set classification accuracy, with the exception of cross-validated OneBitCS, which gives noticeably poorer accuracy. Interestingly, OneBitCS also gives by far the sparsest weight vectors, apparently at the cost of test-error rate. A better tradeoff between test accuracy and weight vector density is given by the EM-tuned GAMP algorithms, which return weight vectors that are about half as dense as those returned by CDN, TRON, TFOCS, and cross-validated GAMP, but that sacrifice only a fraction-of-a-percent in test accuracy.

[9]For OneBitCS, the cross-validation grid included sparsity rates between 0.1% and 15%.

Table VIII also shows a wide range of runtimes. OneBitCS gives by far the fastest post-tuning runtime, for the reasons described earlier. Among the total runtimes, however, the two fastest are EM-GAMP based, with the best (at 105 seconds) beating the fastest high-accuracy non-GAMP algorithm (i.e., TFOCS at 1086 seconds) by more than a factor of 10. That said, some caution must be used when comparing runtimes. For example, while all algorithms were given a "stopping tolerance" of 10^{-3}, the algorithms apply this tolerance in different ways. Also, CDN and TRON are implemented in C++, while GAMP is implemented in object-oriented MATLAB (and therefore is far from optimized).

To understand how performance is impacted in a data-starved regime (i.e., $N > M$), we tested each algorithm on the same RCV1 dataset, but *without* swapping the train/test datasets as was done in [17], [47] and our Table VIII. The results are shown in Table IX. Similar to our other RCV experiment, we see all classifiers yielding very similar test error rates, with the exception of OneBitCS, which does significantly worse. Again, however, OneBitCS generates an extremely sparse weight vector at the expense of test error rate, whereas some the EM-tuned BG-HL and L1-LR GAMP algorithms offer (milder) density reduction without a significant cost in test accuracy. Finally, the two fastest total runtimes are earned by the spGAMP algorithms, and the fastest (BG-PR at 4 seconds) is about 3 times as quick as the fastest high-accuracy non-GAMP algorithm (i.e., CDN at 11 seconds).

Finally, we note that, although GAMP was derived under the assumption that the elements of $\boldsymbol{X}$ are realizations of a an i.i.d sub-Gaussian distribution, it worked well even with the $\boldsymbol{X}$ of this experiment, which was far from i.i.d sub-Gaussian. We attribute the robust performance of GAMP to the "damping" mechanism included in the GAMPmatlab implementation, which was first described in [50] and rigorously analyzed in [34]. Essentially, damping slows down the updates with the goal of preventing divergence.

C. Robust Classification

In Section IV-D, 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 $p^*_{y|z}$. We now evaluate the performance of this robustification method. To

do so, we first generated examples[10] $(y_m, \boldsymbol{x}_m)$ with balanced classes such that the Bayes-optimal classification boundary is a hyper-plane 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, (25), 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 $\alpha = 2\,\mathrm{M}\mu$ [37]. 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 V, 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 (41).

In Fig. 4, 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 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.

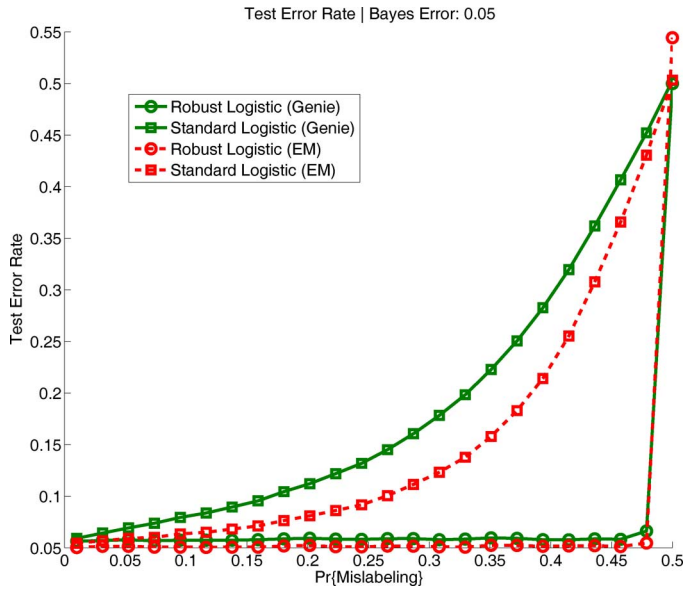

Fig. 4. Test error rate of genie-aided (solid curves) and EM-tuned (dashed curves) instances of standard logistic () and robust logistic (○) classifiers, as a function of mislabeling probability γ, with $M = 8192$, $N = 512$, and Bayes error rate $\varepsilon_{\mathrm{B}} = 0.05$.

D. Multi-Voxel Pattern Analysis

Multi-voxel pattern analysis (MVPA) has become an important tool for analyzing functional MRI (fMRI) data [3]–[5]. Cognitive neuro-scientists, who study how the human brain functions at a physical level, employ MVPA not only 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 often feature selection, not classification.

To investigate the performance of GAMP for MVPA, we conducted an experiment using the well-known Haxby dataset [3]. The Haxby dataset consists of fMRI data collected from 6 subjects with 12 "runs" per subject. In each run, the subject passively viewed blocks of 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 four algorithms: i) ℓ_1-penalized logistic regression (L1-LR) as implemented using cross-validation-tuned TFOCS [48], ii) L1-LR as implemented using EM-tuned max-sum GAMP, iii) sum-product GAMP under a Bernoulli-Laplace prior and logistic activation function (BL-LR), and iv) a cross-validation-tuned OneBitCS [9] classifier.

Algorithm performance (i.e., error-rate, sparsity, and 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 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. A similar two-level cross-validation strategy was applied for selection of the sparsity rate in OneBitCS, using a logarithmically spaced 50-point grid over sparsity rates between 0.1%

[10]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}.

TABLE X
PERFORMANCE OF CROSS-VALIDATION TUNED L1-LR TFOCS ("TFOCS"), EM-TUNED L1-LR MAX-SUM GAMP ("L1-LR"), EM-TUNED BG-LR SUM-PRODUCT GAMP ("BG-LR"), AND CROSS-VALIDATION TUNED ONE BITCS ("1-BIT") CLASSIFIERS ON VARIOUS HAXBY PAIRWISE COMPARISONS

	Error Rate (%)				Sparsity (%)				Consistency (%)				Runtime (s)			
Comparison	TFOCS	L1-LR	BG-LR	1-Bit	TFOCS	L1-LR	BG-LR	1-Bit	TFOCS	L1-LR	BG-LR	1-Bit	TFOCS	L1-LR	BG-LR	1-Bit
Cat vs. Scissors	9.7	11.1	9.3	**5.1**	0.1	0.07	**0.01**	0.12	38	43	**60**	57	1318	137	**21**	202
Cat vs. Shoe	**6.1**	**6.1**	11.6	6.5	0.14	0.07	**0.01**	0.12	34	47	**60**	59	1347	191	**24**	205
Cat vs. House	0.4	**0.0**	1.4	3.7	0.04	0.02	**0.01**	0.12	53	**87**	84	75	1364	144	**18**	202
Bottle vs. Shoe	29.6	30.5	23.6	**20.4**	0.2	0.1	**0.01**	0.12	23	31	36	**53**	1417	166	**22**	205
Bottle vs. Chair	**13.9**	**13.9**	15.7	26.9	0.1	0.07	**0.01**	0.12	30	45	**61**	37	1355	150	**21**	203
Face vs. Chair	**0.9**	**0.9**	6.9	2.8	0.09	0.05	**0.01**	0.12	43	67	68	**76**	1362	125	**24**	205
Average	**10.1**	10.4	11.4	10.9	0.11	0.06	**0.01**	0.12	37	53	**62**	60	1358	152	**22**	204

and 15%. 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 V to the 11-fold training data.

Table X reports the results of the above-described experiment for six pairwise comparisons. For all but BG-LR GAMP, sparsity refers to the average percentage of non-zero elements in the learned weight vectors. But, since BG-LR GAMP's weights are non-zero with probability one, we instead define BG-LR's sparsity as the number of weights with posterior probability $p(w_n \neq 0|\boldsymbol{y}) > 1/2$, as we did with the other sum-product-GAMP classifiers in earlier experiments. Consistency refers to the average Jaccard index between weight-vector supports, i.e.,

$$\text{consistency} \triangleq \frac{1}{12}\sum_{i=1}^{12}\frac{1}{11}\sum_{j\neq i}\frac{|\mathcal{S}_i \cap \mathcal{S}_j|}{|\mathcal{S}_i \cup \mathcal{S}_j|} \tag{42}$$

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 quickly computes weight vectors with low estimated error rate, high consistency, and relatively low density. It should be emphasized that minimizing estimated error rate alone is not of sole importance, especially for this dataset, where the total number of samples is so few that the error rate estimates are understood to be very noisy. Moreover, since the goal of MVPA is to identify which voxels of the brain are most important in discriminating between cognitive states, consistency among folds *is* very important.

Unfortunately, Table X reveals no clear winner among the algorithms under test. Starting with the estimated error rates, all four algorithms yielded similar comparison-averaged rates, spanning the range from 10.1% (for TFOCS) to 11.4% (for BG-LR GAMP). Interestingly, the algorithm ranking under the consistency metric was exactly the opposite of that for the error-rate metric: BG-LR GAMP yielded the highest consistency (of 62%) and TFOCS the lowest consistency (of 37%). In terms of sparsity, BG-LR GAMP appears to be the winner, but perhaps a direct comparison to the other algorithms should be avoided due to the differences in the definition of sparsity. For runtime, however, the clear winner is EM-tuned BG-LR GAMP, which runs an order-of-magnitude faster than cross-validated OneBitCS and nearly two orders-of-magnitude faster than cross-validated TFOCS.

A direct comparison between cross-validated TFOCS and EM-tuned L1-LR GAMP is illuminating, since these two algorithms share the L1-LR objective and thus differ mainly in tuning strategy.[11] For this Haxby data, Table X shows that L1-LR GAMP's classifiers are uniformly more sparse (and nearly twice as sparse on average) as those generated by TFOCS, while suffering only a small degradation in error-rate. Meanwhile, L1-LR GAMP's classifiers are uniformly more consistent, and its runtimes are about 9 × faster on average.

VII. CONCLUSION

In this work, we presented the first comprehensive study of the *generalized approximate message passing* (GAMP) algorithm [22] in the context of linear binary classification and feature selection. We established that a number of popular discriminative models, including logistic and probit regression, as well as support vector machines (via hinge loss), 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 hyper-parameters that govern our probabilistic models adaptively from the data using expectation-maximization (EM), a trait which can be advantageous in terms of runtime. 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.

In a numerical study, we confirmed the efficacy of our approach on both synthetic and real-world 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 GAMP can attain nearly optimal error rates in the $N \gg M$ regime when N is sufficiently large and the number of discriminatory features, K is sufficiently small. Furthermore, we observed that the robust classification extension can substantially outperform a non-robust counterpart.

APPENDIX
SUM-PRODUCT GAMP HINGE-LOSS COMPUTATIONS

In this appendix, we describe the steps needed to compute the sum-product GAMP nonlinear steps for the hinge-loss acti-

[11] It is known that, if max-sum GAMP converges, then it converges to a critical point of the optimization objective [33], which in the (convex) L1-LR case is unique.

vation function, (28). For convenience, we define the associated *un-normalized* likelihood function

$$\tilde{p}_{\mathsf{y}|\mathsf{z}}(y|z) \triangleq \exp\big(-\max(0, 1-yz)\big), \quad y \in \{-1, 1\}. \quad (43)$$

Note from (9) that the sum-product $(\hat{z}, \tau_z)$ can be interpreted as the posterior mean and variance of a random variable, z, with prior $\mathcal{N}(\hat{p}, \tau_p)$ and likelihood proportional to $\tilde{p}_{\mathsf{y}|\mathsf{z}}(y|z)$.

To compute the statistics $\hat{z} \equiv \mathrm{E}[\mathsf{z}|\mathsf{y} = y]$ and $\tau_z \equiv \mathrm{var}\{\mathsf{z}|\mathsf{y} = y\}$, we first write the posterior pdf as

$$p_{\mathsf{z}|\mathsf{y}}(z|y) = C_y^{-1} \tilde{p}_{\mathsf{y}|\mathsf{z}}(y|z) p_{\mathsf{z}}(z), \quad (44)$$

where C_y is an appropriate normalization constant. Defining

$$\alpha_y \triangleq ((1-\tau_p) - y\hat{p})/\sqrt{\tau_p} \quad (45)$$

$$\beta_y \triangleq (y\hat{p} - 1)/\sqrt{\tau_p} \quad (46)$$

$$\delta_y \triangleq y\hat{p} - 1 + \tau_p/2, \quad (47)$$

it can be shown [36] that

$$C_1 = \int_{-\infty}^{1} \exp(z-1)\mathcal{N}(z; \hat{p}, \tau_p) + \int_{1}^{\infty} \mathcal{N}(z; \hat{p}, \tau_p) \quad (48)$$

$$= \exp(\delta_1)\Phi(\alpha_1) + \Phi(\beta_1) \quad (49)$$

The posterior mean for $y = 1$ is therefore given by

$$\mathrm{E}[\mathsf{z}|\mathsf{y} = 1]$$

$$= \frac{1}{C_1} \int_z z\, \tilde{p}_{\mathsf{y}|\mathsf{z}}(y = 1|z) p_{\mathsf{z}}(z) \quad (50)$$

$$= \frac{1}{C_1} \left[e^{\delta_1} \int_{-\infty}^{1} z\mathcal{N}(z; \hat{p} + \tau_p, \tau_p) + \int_{1}^{\infty} z\mathcal{N}(z; \hat{p}, \tau_p) \right] \quad (51)$$

$$= \frac{e^{\delta_1}\Phi(\alpha_1)}{C_1} \int_{-\infty}^{1} z \frac{\mathcal{N}(z; \hat{p} + \tau_p, \tau_p)}{\Phi(\alpha_1)} + \frac{\Phi(\beta_1)}{C_1} \int_{1}^{\infty} z \frac{\mathcal{N}(z; \hat{p}, \tau_p)}{\Phi(\beta_1)}, \quad (52)$$

where each integral in (52) represents the first moment of a truncated normal random variable. Similar expressions can be derived for $\mathrm{E}[\mathsf{z}|\mathsf{y} = -1]$. Then, defining the quantities

$$\gamma_y \triangleq e^{-\delta_y}\Phi(\beta_y)/\Phi(\alpha_y) \quad (53)$$

$$\underline{\mu}_y \triangleq \hat{p} + y\big(\tau_p - \sqrt{\tau_p}\phi(\alpha_y)/\Phi(\alpha_y)\big) \quad (54)$$

$$\bar{\mu}_y \triangleq \hat{p} + y\sqrt{\tau_p}\phi(\beta_y)/\Phi(\beta_y), \quad (55)$$

it can be shown [51] that, for $y \in \{-1, 1\}$,

$$\hat{z}(y) = \mathrm{E}[\mathsf{z}|\mathsf{y} = y] = (1+\gamma_y)^{-1}\underline{\mu}_y + (1+\gamma_y^{-1})^{-1}\bar{\mu}_y. \quad (56)$$

To compute $\tau_z \equiv \mathrm{var}\{\mathsf{z}|\mathsf{y} = y\}$, it suffices to derive an expression for $\mathrm{E}[\mathsf{z}^2|\mathsf{y} = y]$. Following the same line of reasoning that produced (52), we find

$$\mathrm{E}[\mathsf{z}^2|\mathsf{y} = 1] = \frac{e^{\delta_1}\Phi(\alpha_1)}{C_1} \int_{-\infty}^{1} z^2 \frac{\mathcal{N}(z; \hat{p} + \tau_p, \tau_p)}{\Phi(\alpha_1)} + \frac{\Phi(\beta_1)}{C_1} \int_{1}^{\infty} z^2 \frac{\mathcal{N}(z; \hat{p}, \tau_p)}{\Phi(\beta_1)}, \quad (57)$$

where each integral in (57) is the second moment of a truncated normal random variable. A similar expression can be derived for $\mathrm{E}[\mathsf{z}^2|\mathsf{y} = -1]$. Defining

$$\underline{v}_y \triangleq \tau_p \left[1 - \frac{\phi(\alpha_y)}{\Phi(\alpha_y)} \left(\frac{\phi(\alpha_y)}{\Phi(\alpha_y)} + \alpha_y \right) \right] \quad (58)$$

$$\bar{v}_y \triangleq \tau_p \left[1 - \frac{\phi(\beta_y)}{\Phi(\beta_y)} \left(\frac{\phi(\beta_y)}{\Phi(\beta_y)} + \beta_y \right) \right], \quad (59)$$

it can be shown [51] that

$$\mathrm{E}[\mathsf{z}^2|\mathsf{y} = y] = (1+\gamma_y)^{-1}(\underline{v}_y + \underline{\mu}_y^2) + (1+\gamma_y^{-1})^{-1}(\bar{v}_y + \bar{\mu}_y^2), \quad (60)$$

allowing us to compute $\tau_z(y) = \mathrm{E}[\mathsf{z}^2|\mathsf{y} = y] - \hat{z}^2(y)$.

REFERENCES

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

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

[3] 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," *Sci.*, vol. 293, pp. 2425–2430, Sep. 2001.

[4] K. A. Norman, S. M. Polyn, G. J. Detre, and J. V. Haxby, "Beyond mind-reading: Multi-voxel pattern analysis of fMRI data," *Trends Cognit. Sci.*, vol. 10, pp. 424–430, Sep. 2006.

[5] F. Pereira, T. Mitchell, and M. Botvinick, "Machine learning classifiers and fMRI: A tutorial overview," *NeuroImage*, vol. 45, pp. S199–S209, Mar. 2009.

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

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

[8] P. T. Boufounos and R. G. Baraniuk, "1-bit compressive sensing," presented at the Conf. Inf. Sci. Sys., Princeton, NJ, USA, Mar. 2008.

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

[10] D. Koller and M. Sahami, L. Saitta, Ed., "Toward optimal feature selection," in *Proc. 13th Int. Conf. Mach. Learn. (ICML)*, Bari, Italy, 1996, pp. 284–292.

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

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

[13] M. Figueiredo, "Adaptive sparseness using Jeffreys' prior," in *Proc. 14th Conf. Adv. Neural Inf. Process. Syst.*, Cambridge, MA, USA, 2001, pp. 697–704.

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

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

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

[17] 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.

[18] A. Gupta, R. Nowak, and B. Recht, "Sample complexity for 1-bit compressed sensing and sparse classification," presented at the Int. Symp. Inf. Theory (ISIT), Austin, TX, 2010.

[19] 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.

[20] 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.

[21] U. S. Kamilov, V. K. Goyal, and S. Rangan, "Message-passing de-quantization with applications to compressed sensing," *IEEE Trans. Signal Process.*, vol. 60, pp. 6270–6281, Dec. 2012.
[22] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Int. Symp. Inf. Theory*, St. Petersburg, Russia, Aug. 2011, pp. 2168–2172.
[23] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," in *Proc. Nat. Acad. Sci.*, Nov. 2009, vol. 106, pp. 18914–18919.
[24] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. Motivation and construction," in *Proc. Inf. Theory Workshop*, Jan. 2010, pp. 1–5.
[25] A. Javanmard and A. Montanari, "State evolution for general approximate message passing algorithms, with applications to spatial coupling," *Inf. Inference*, vol. 2, no. 2, pp. 115–144, 2013.
[26] J. Ziniel, S. Rangan, and P. Schniter, "A generalized framework for learning and recovery of structured sparse signals," presented at the IEEE Statist. Signal Process. Workshop, Ann Arbor, MI, USA, Aug. 2012.
[27] J. P. Vila and P. Schniter, "Expectation-Maximization Gaussian-mixture approximate message passing," *IEEE Trans. Signal Process.*, vol. 61, pp. 4658–4672, Oct. 2013.
[28] P. Schniter, "Turbo reconstruction of structured sparse signals," in *Proc. Conf. Inf. Sci. Syst. (CISS)*, Princeton, NJ, USA, Mar. 2010, pp. 1–6.
[29] B. J. Frey and D. J. C. MacKay, "A revolution: Belief propagation in graphs with cycles," *Adv. Neural Inf. Process. Syst.*, pp. 479–485, 1998.
[30] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498–519, Feb. 2001.
[31] R. J. McEliece, D. J. C. MacKay, and J. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *IEEE J. Sel. Areas Commun.*, vol. 16, pp. 140–152, Feb. 1998.
[32] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *Int. . J. Comput. Vis.*, vol. 40, pp. 25–47, Oct. 2000.
[33] S. Rangan, P. Schniter, E. Riegler, A. K. Fletcher, and V. Cevher, "Fixed points of generalized approximate message passing with arbitrary matrices," in *Proc. IEEE Int. Symp. Inf. Theory*, Istanbul, Jul. 2013, pp. 664–668.
[34] S. Rangan, P. Schniter, and A. Fletcher, "On the convergence of generalized approximate message passing with arbitrary matrices," in *Proc. IEEE Int. Symp. Inf. Theory*, Honolulu, HI, USA, Jul. 2014, pp. 236–240.
[35] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, pp. 28–41, Jan. 2004.
[36] J. Ziniel, "Message passing approaches to compressive inference under structured signal priors," Ph.D. dissertation, The Ohio State Univ., Columbus, OH, USA, 2014.
[37] M. I. Jordan, "Why the logistic function? A tutorial discussion on probabilities and neural networks," MIT Comput. Congnit. Sci., Cambridge, MA, USA, Tech. Rep. 9503, Aug. 13, 1995.
[38] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, ISA: MIT Press, 2006.
[39] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song, "Dimensionality reduction via sparse support vector machines," *J. Mach. Learn. Res.*, vol. 3, pp. 1229–1243, 2003.
[40] M. Opper and O. Winther, *Gaussian Processes and SVM: Mean Field Results and Leave-One-Out Estimator*. Cambridge, MA, USA: MIT Press, 2000, ch. 17, pp. 311–326.
[41] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Statist. Soc., B*, vol. 67, no. 2, pp. 301–320, 2005.
[42] A. K. Nigam, K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Mach. Learn.*, vol. 39, pp. 103–134, 2000.
[43] J. P. 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.
[44] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc., B*, vol. 39, pp. 1–38, 1977.
[45] 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 Inf. Process. Syst. Conf.*, Lake Tahoe, NV, USA, Dec. 2012, pp. 2447–2455.
[46] 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.
[47] C. Lin, R. C. Weng, and S. S. Keerthi, "Trust region Newton methods for large-scale logistic regression," in *Proc. 24th Int. Conf. Mach. Learn.*, Corvallis, OR, USA, 2007, pp. 561–568.
[48] S. R. Becker, E. J. Candès, and M. C. Grant, "Templates for convex cone problems with applications to sparse signal recovery," *Math. Prog. Comput.*, vol. 3, no. 3, pp. 165–218, 2011.
[49] C. J. Lin and J. J. Moré, "Newton's method for large-scale bound constrained problems," *SIAM J. Optim.*, vol. 9, pp. 1100–1127, 1999.
[50] P. Schniter and S. Rangan, "Compressive phase retrieval via generalized approximate message passing," presented at the Allerton Conf. Commun., Control, Comput., Monticello, IL, USA, Oct. 2012.
[51] D. R. Barr and E. T. Sherrill, "Mean and variance of truncated normal distributions," *Amer. Statist.*, vol. 53, Nov. 1999.

Justin Ziniel (S'11) received the B.S., M.S., and Ph.D. degrees in Electrical and Computer Engineering from The Ohio State University in Columbus, OH, in 2007, 2012, and 2014, respectively.

His research interests include statistical signal processing, machine learning, and interdisciplinary "big data" inference problems.

Philip Schniter (F'14) received the B.S. and M.S. degrees in Electrical Engineering from the University of Illinois at Urbana-Champaign in 1992 and 1993, respectively, and the Ph.D. degree in Electrical Engineering from Cornell University in Ithaca, NY, in 2000.

From 1993 to 1996 he was employed by Tektronix Inc. in Beaverton, OR as a systems engineer. After receiving the Ph.D. degree, he joined the Department of Electrical and Computer Engineering at The Ohio State University, Columbus, where he is currently a Professor and a member of the Information Processing Systems (IPS) Lab. In 2008–2009 he was a visiting professor at Eurecom, Sophia Antipolis, France, and Supélec, Gif-sur-Yvette, France.

In 2003, Dr. Schniter received the National Science Foundation CAREER Award. His areas of interest currently include statistical signal processing, wireless communications and networks, and machine learning.

Per Sederberg, photograph and biography not available at the time of publication.