Sparse Multinomial Logistic Regression via Approximate Message Passing

Evan Byrne and Philip Schniter, *Fellow, IEEE*

***Abstract*—For the problem of multi class linear classification and feature selection, we propose approximate message passing approaches to sparse multinomial logistic regression (MLR). First, we propose two algorithms based on the Hybrid Generalized Approximate Message Passing framework: one finds the maximum *a posteriori* linear classifier and the other finds an approximation of the test-error-rate minimizing linear classifier. Then we design computationally simplified variants of these two algorithms. Next, we detail methods to tune the hyperparameters of their assumed statistical models using Stein's unbiased risk estimate and expectation-maximization, respectively. Finally, using both synthetic and real-world datasets, we demonstrate improved error-rate and runtime performance relative to existing state-of-the-art approaches to sparse MLR.**

***Index Terms*—Classification, feature selection, multinomial logistic regression (MLR), belief propagation, approximate message passing.**

I. Introduction

A. Objective

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

$$\widehat{y}_0 = \arg\max_d \, [\boldsymbol{z}_0]_d. \tag{1}$$

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

We are particularly interested in the setting where the number of features, N, is large and greatly exceeds the number of training examples, M. Such problems arise in a number of important applications, such as micro-array gene expression [1], [2], multi-voxel pattern analysis (MVPA) [3], [4], text mining [5], [6], and analysis of marketing data [7].

In the $N \gg M$ case, accurate linear classification and feature selection may be possible if the labels are influenced by a sufficiently small number, K, of the total N features. For example, in binary linear classification, performance guarantees are possible with only $M = O(K \log N/K)$ training examples when $\boldsymbol{a}_m$ is i.i.d. Gaussian [8].

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

B. Multinomial Logistic Regression

For multiclass linear classification and feature selection, we focus on the approach known as multinomial logistic regression (MLR) [9], which can be described using a generative probabilistic model. Here, the label vector $\boldsymbol{y} \triangleq [y_0, \ldots, y_M]^\mathsf{T}$ is modeled as a realization of a random[1] vector $\mathsf{y} \triangleq [\mathsf{y}_0, \ldots, \mathsf{y}_M]^\mathsf{T}$, the "true" weight matrix $\boldsymbol{X}$ is modeled as a realization of a random matrix X, and the features $\boldsymbol{A} \triangleq [\boldsymbol{a}_0, \ldots, \boldsymbol{a}_M]^\mathsf{T}$ are treated as deterministic. Moreover, the labels y_m are modeled as conditionally independent given the scores $\mathsf{z}_m \triangleq \mathsf{X}^\mathsf{T} \boldsymbol{a}_m$, i.e.,

$$\Pr\{\mathsf{y} = \boldsymbol{y} \,|\, \mathsf{X} = \boldsymbol{X}; \boldsymbol{A}\} = \prod_{m=1}^M p_{\mathsf{y}|\mathsf{z}}(y_m | \boldsymbol{X}^\mathsf{T} \boldsymbol{a}_m), \tag{2}$$

and distributed according to the multinomial logistic (or softmax) pmf:

$$p_{\mathsf{y}|\mathsf{z}}(y_m | \boldsymbol{z}_m) = \frac{\exp([\boldsymbol{z}_m]_{y_m})}{\sum_{d=1}^D \exp([\boldsymbol{z}_m]_d)}, \; y_m \in \{1, \ldots, D\}. \tag{3}$$

The rows x_n^T of the weight matrix X are then modeled as i.i.d.,

$$p_{\mathsf{X}}(\boldsymbol{X}) = \prod_{n=1}^N p_{\mathsf{x}}(\boldsymbol{x}_n), \tag{4}$$

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

C. Existing Methods

Several sparsity-promoting MLR algorithms have been proposed (e.g., [10]–[15]), differing in their choice of p_{x} and methodology of estimating X. For example, [11]–[13] use the i.i.d. Laplacian prior

$$p_{\mathsf{x}}(\boldsymbol{x}_n; \lambda) = \prod_{d=1}^D \frac{\lambda}{2} \exp(-\lambda |x_{nd}|), \tag{5}$$

Manuscript received September 15, 2015; revised June 14, 2016; accepted July 4, 2016. Date of publication July 21, 2016; date of current version September 8, 2016. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Xiao-Ping Zhang. This work was supported in part by the National Science Foundation under Grant CCF-1018368 and Grant CCF-1218754.

The authors are with the Department of Electrical and Computer Engineering, Ohio State University, Columbus, OH 43210 USA (e-mail: byrne.133@osu.edu; schniter@ece.osu.edu).

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.2016.2593691

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

with λ tuned via cross-validation (CV). To circumvent this tuning problem, [14] employs the Laplacian scale mixture

$$p_{\mathsf{x}}(\boldsymbol{x}_n) = \prod_{d=1}^{D} \int \left[\frac{\lambda}{2}\exp(-\lambda|x_{nd}|)\right] p(\lambda)\,\mathrm{d}\lambda, \tag{6}$$

with Jeffrey's non-informative hyperprior $p(\lambda) \propto \frac{1}{\lambda}1_{\lambda \geq 0}$. The relevance vector machine approach [10] uses the Gaussian scale mixture

$$p_{\mathsf{x}}(\boldsymbol{x}_n) = \prod_{d=1}^{D} \int \mathcal{N}(x_{nd}; 0, \nu) p(\nu)\,\mathrm{d}\nu, \tag{7}$$

with inverse-gamma $p(\nu)$ (i.e., the conjugate hyperprior), resulting in an i.i.d. student's t distribution for p_{x}. However, other choices are possible. For example, the exponential hyperprior $p(\nu; \lambda) = \frac{\lambda^2}{2}\exp(-\frac{\lambda^2}{2}\nu)1_{\nu \geq 0}$ would lead back to the i.i.d. Laplacian distribution (5) for p_{x} [16]. Finally, [15] uses

$$p_{\mathsf{x}}(\boldsymbol{x}_n; \lambda) \propto \exp(-\lambda\|\boldsymbol{x}_n\|_2), \tag{8}$$

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

Once the probabilistic model (2)–(4) has been specified, a procedure is needed to infer the weights $\boldsymbol{\mathsf{X}}$ from the training data $\{(y_m, \boldsymbol{a}_m)\}_{m=1}^{M}$. The Laplacian-prior methods [11]–[13], [15] use the maximum *a posteriori* (MAP) estimation framework:

$$\widehat{\boldsymbol{X}} = \arg\max_{\boldsymbol{X}} \log p(\boldsymbol{X}|\boldsymbol{y}; \boldsymbol{A}) \tag{9}$$

$$= \arg\max_{\boldsymbol{X}} \sum_{m=1}^{M} \log p_{\mathsf{y}|\boldsymbol{\mathsf{z}}}(y_m|\boldsymbol{X}^{\mathsf{T}}\boldsymbol{a}_m) + \sum_{n=1}^{N} \log p_{\mathsf{x}}(\boldsymbol{x}_n), \tag{10}$$

where Bayes' rule was used for (10). Under p_{x} from (5) or (8), the second term in (10) reduces to $-\lambda\sum_{n=1}^{N}\|\boldsymbol{x}_n\|_1$ or $-\lambda\sum_{n=1}^{N}\|\boldsymbol{x}_n\|_2$, respectively. In this case, (10) is concave and can be maximized in polynomial time; [11]–[13], [15] employ (block) coordinate ascent for this purpose. The papers [10] and [14] handle the scale-mixture priors (6) and (7), respectively, using the evidence maximization framework [17]. This approach yields a double-loop procedure: the hyperparameter λ or ν is estimated in the outer loop, and—for fixed λ or ν—the resulting concave (i.e., ℓ_2 or ℓ_1 regularized) MAP optimization is solved in the inner loop.

The methods [10]–[15] described above all yield a sparse point estimate $\widehat{\boldsymbol{X}}$. Thus, feature selection is accomplished by examining the row-support of $\widehat{\boldsymbol{X}}$ and classification is accomplished through (1).

D. Contributions

In Section II, we propose new approaches to sparse-weight MLR based on the *hybrid generalized approximate message passing* (HyGAMP) framework from [18]. HyGAMP offers tractable approximations of the sum-product and min-sum message passing algorithms [19] by leveraging results of the central limit theorem that hold in the large-system limit: $\lim_{N,M\to\infty}$ with fixed N/M. Without approximation, both the sum-product algorithm (SPA) and min-sum algorithm (MSA) are intractable due to the forms of $p_{\mathsf{y}|\boldsymbol{\mathsf{z}}}$ and p_{x} in our problem.

For context, we note that HyGAMP is a generalization of the original GAMP approach from [20], which cannot be directly applied to the MLR problem because the likelihood function (3) is not separable, i.e., $p_{\mathsf{y}|\boldsymbol{\mathsf{z}}}(y_m|\boldsymbol{z}_m) \neq \prod_d p(y_m|z_{md})$. GAMP can, however, be applied to *binary* classification and feature selection, as in [21]. Meanwhile, GAMP is itself a generalization of the original AMP approach from [22], [23], which requires $p_{\mathsf{y}|\boldsymbol{\mathsf{z}}}$ to be both separable and Gaussian.

With the HyGAMP algorithm from [18], message passing for sparse-weight MLR reduces to an iterative update of $O(M + N)$ multivariate Gaussian pdfs, each of dimension D. Although HyGAMP makes MLR tractable, it is still not computationally practical for the large values of M and N in contemporary applications (e.g., $N \sim 10^4$ to 10^6 in genomics and MVPA). Similarly, the non-conjugate variational message passing technique from [24] requires the update of $O(MN)$ multivariate Gaussian pdfs of dimension D, which is even less practical for large M and N.

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

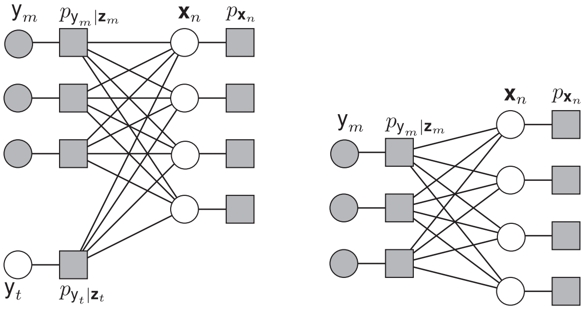
In Section IV, we detail two approaches to tune the hyperparameters that control the statistical models assumed by SHyGAMP, one based on the expectation-maximization (EM) methodology from [25] and the other based on a variation of the Stein's unbiased risk estimate (SURE) methodology from [26]. We also give numerical evidence that these methods yield near-optimal hyperparameter estimates.

Finally, in Section V, we compare our proposed SHyGAMP methods to the state-of-the-art MLR approaches [13], [14] on both synthetic and practical real-world problems. Our experiments suggest that our proposed methods offer simultaneous improvements in classification error rate and runtime.

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

II. HyGAMP for Multiclass Classification

In this section, we detail the application of HyGAMP [18] to multiclass linear classification. In particular, we show that the



(a) Full graph (b) Reduced graph

Fig. 1. Factor graph representations of (14), with white/gray circles denoting unobserved/observed random variables, and gray rectangles denoting pdf "factors". (a) Full graph. (b) Reduced graph.

SPA variant of HyGAMP is a loopy belief propagation (LBP) approximation of the classification-error-rate minimizing linear classifier and that the MSA variant is an LBP approach to solving the MAP problem (10).

A. Classification via Sum-Product HyGAMP

Suppose that we are given M labeled training pairs $\{(y_m, \boldsymbol{a}_m)\}_{m=1}^{M}$ and T test feature vectors $\{\boldsymbol{a}_t\}_{t=M+1}^{M+T}$ associated with unknown test labels $\{\mathsf{y}_t\}_{t=M+1}^{M+T}$, all obeying the MLR statistical model (2)–(4). Consider the problem of computing the classification-error-rate minimizing hypotheses $\{\widehat{y}_t\}_{t=M+1}^{M+T}$,

$$\widehat{y}_t = \arg \max_{y_t \in \{1,\ldots,D\}} p_{\mathsf{y}_t|\mathbf{y}_{1:M}}\big(y_t \,\big|\, \boldsymbol{y}_{1:M}; \boldsymbol{A}\big), \tag{11}$$

under known $p_{\mathsf{y}|\mathsf{z}}$ and $p_{\mathbf{x}}$, where $\boldsymbol{y}_{1:M} \triangleq [y_1, \ldots, y_M]^{\mathsf{T}}$ and $\boldsymbol{A} \triangleq [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{M+T}]^{\mathsf{T}}$. The probabilities in (11) can be computed via the marginalization

$$p_{\mathsf{y}_t|\mathbf{y}_{1:M}}\big(y_t \,\big|\, \boldsymbol{y}_{1:M}; \boldsymbol{A}\big) = p_{\mathsf{y}_t,\mathbf{y}_{1:M}}\big(y_t, \boldsymbol{y}_{1:M}; \boldsymbol{A}\big) Z_{\mathbf{y}}^{-1} \tag{12}$$

$$= Z_{\mathbf{y}}^{-1} \sum_{\boldsymbol{y} \in \mathcal{Y}_t(y_t)} \int p_{\mathbf{y},\mathbf{X}}(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{A})\, \mathrm{d}\boldsymbol{X}, \tag{13}$$

with scaling constant $Z_{\mathbf{y}}^{-1}$, label vector $\boldsymbol{y} = [y_1, \ldots, y_{M+T}]^{\mathsf{T}}$, and constraint set $\mathcal{Y}_t(y) \triangleq \big\{\widetilde{\boldsymbol{y}} \in \{1, \ldots, D\}^{M+T} \text{s.t. } [\widetilde{\boldsymbol{y}}]_t = y \text{ and } [\widetilde{\boldsymbol{y}}]_m = y_m \,\forall m = 1, \ldots, M\big\}$, which fixes the tth element of $\boldsymbol{y}$ at the value y and the first M elements of $\boldsymbol{y}$ at the values of the corresponding training labels. Due to (2) and (4), the joint pdf in (13) factors as

$$p_{\mathbf{y},\mathbf{X}}(\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{A}) = \prod_{m=1}^{M+T} p_{\mathsf{y}|\mathbf{z}}(y_m \mid \boldsymbol{X}^{\mathsf{T}} \boldsymbol{a}_m) \prod_{n=1}^{N} p_{\mathbf{x}}(\boldsymbol{x}_n). \tag{14}$$

The factorization in (14) is depicted by the *factor graph* in Fig. 1(a), where the random variables $\{\mathsf{y}_m\}$ and random vectors $\{\mathbf{x}_n\}$ are connected to the pdf factors in which they appear.

Since exact computation of the marginal posterior test-label probabilities is an NP-hard problem [27], we are interested in alternative strategies, such as those based on LBP by the SPA [19]. Although a direct application of the SPA is itself intractable when $p_{\mathsf{y}|\mathbf{z}}$ takes the MLR form (3), the SPA simplifies in the large-system limit under i.i.d. sub-Gaussian $\boldsymbol{A}$, leading to the HyGAMP approximation [18] given[2] in Algorithm 1. Although in practical MLR applications $\boldsymbol{A}$ is not i.i.d. Gaussian,[3] the numerical results in Section V suggest that treating it as such works sufficiently well.

We note from Fig. 1(a) that the HyGAMP algorithm is applicable to a factor graph with vector-valued variable nodes. As such, it generalizes the GAMP algorithm from [20], which applies only to a factor graph with scalar-variable nodes. Below, we give a brief explanation for the steps in Algorithm 1. For those interested in more details, we suggest [18] for an overview and derivation of HyGAMP, [20] for an overview and derivation of GAMP, [28] for rigorous analysis of GAMP under large i.i.d. sub-Gaussian $\boldsymbol{A}$, and [29], [30] for fixed-point and local-convergence analysis of GAMP under arbitrary $\boldsymbol{A}$.

Lines 6 and 7 of Algorithm 1 produce an approximation of the posterior mean and covariance of $\mathbf{x}_n$ at each iteration t. Similarly, lines 15 and 16 produce an approximation of the posterior mean and covariance of $\mathbf{z}_m \triangleq \mathbf{X}^{\mathsf{T}} \boldsymbol{a}_m$. The posterior mean and covariance of $\mathbf{x}_n$ are computed from the intermediate quantity $\widehat{\boldsymbol{r}}_n(t)$, which behaves like a noisy measurement of the true $\boldsymbol{x}_n$. In particular, for i.i.d. Gaussian $\boldsymbol{A}$ in the large-system limit, $\widehat{\boldsymbol{r}}_n(t)$ is a typical realization of the random vector $\mathbf{r}_n = \boldsymbol{x}_n + \mathbf{v}_n$ with $\mathbf{v}_n \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_n^{\mathbf{r}}(t))$. Thus, the approximate posterior pdf used in lines 6 and 7 is

$$p_{\mathbf{x}|\mathbf{r}}(\boldsymbol{x}_n | \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}_n^{\mathbf{r}}) = \frac{p_{\mathbf{x}}(\boldsymbol{x}_n) \mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}_n^{\mathbf{r}})}{\int p_{\mathbf{x}}(\boldsymbol{x}'_n) \mathcal{N}(\boldsymbol{x}'_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}_n^{\mathbf{r}})\, \mathrm{d}\boldsymbol{x}'_n}. \tag{15}$$

A similar interpretation holds for HyGAMP's approximation of the posterior mean and covariance of $\mathbf{z}_m$ in lines 15 and 16, which uses the intermediate vector $\widehat{\boldsymbol{p}}_m(t)$ and the approximate posterior pdf

$$\begin{aligned} &p_{\mathbf{z}|\mathsf{y},\mathbf{p}}(\boldsymbol{z}_m | y_m, \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}_m^{\mathbf{p}}) \\ &= \frac{p_{\mathsf{y}|\mathbf{z}}(y_m | \boldsymbol{z}_m) \mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}_m^{\mathbf{p}})}{\int p_{\mathsf{y}|\mathbf{z}}(y_m | \boldsymbol{z}'_m) \mathcal{N}(\boldsymbol{z}'_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}_m^{\mathbf{p}})\, \mathrm{d}\boldsymbol{z}'_m}. \end{aligned} \tag{16}$$

B. Classification via Min-Sum HyGAMP

As discussed in Section I-C, an alternative approach to linear classification and feature selection is through MAP estimation of the true weight matrix $\boldsymbol{X}$. Given a likelihood of the form (2) and a prior of the form (4), the MAP estimate is the solution to the optimization problem (10).

Similar to how the SPA can be used to compute approximate marginal posteriors in loopy graphs, the MSA [19] can be used to compute the MAP estimate. Although a direct application of the MSA is intractable when $p_{\mathsf{y}|\mathbf{z}}$ takes the MLR form (3), the MSA simplifies in the large-system limit under i.i.d. sub-Gaussian $\boldsymbol{A}$, leading to the MSA form of HyGAMP specified in Algorithm 1.

As described in Section II-A, when $\boldsymbol{A}$ is large and i.i.d. sub-Gaussian, the vector $\widehat{\boldsymbol{r}}_n(t)$ in Algorithm 1 behaves like a

[2] The HyGAMP algorithm in [18] is actually more general than what is specified in Algorithm 1, but the version in Algorithm 1 is sufficient to handle the factor graph in Fig. 1(a).

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

Algorithm 1: HyGAMP.

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

Gaussian-noise-corrupted observation of the true $\boldsymbol{x}_n$ with noise covariance $\boldsymbol{Q}^{\mathsf{r}}_n(t)$. Thus, line 3 can be interpreted as MAP estimation of $\boldsymbol{x}_n$ and line 4 as measuring the local curvature of the corresponding MAP cost. Similar interpretations hold for MAP estimation of $\boldsymbol{z}_m$ via lines 12 and 13.

C. Implementation of Sum-Product HyGAMP

From Algorithm 1, we see that HyGAMP requires inverting $M + N$ matrices of size $D \times D$ (for lines 18 and 20) in addition to solving $M + N$ joint inference problems of dimension D in lines 3–7 and 12–16. We now briefly discuss the latter problems for the sum-product version of HyGAMP.

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

$$p_{\mathsf{x}}(\boldsymbol{x}_n) = (1 - \beta)\delta(\boldsymbol{x}_n) + \beta \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{0}, v\boldsymbol{I}), \tag{17}$$

where $\delta(\cdot)$ denotes the Dirac delta and $\beta \in (0, 1]$. In this case, it can be shown [31] that the mean and variance computations in lines 6–7 of Algorithm 1 reduce to

$$C_n = 1 + \frac{1-\beta}{\beta} \frac{\mathcal{N}(\boldsymbol{0}; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}^{\mathsf{r}}_n)}{\mathcal{N}(\boldsymbol{0}; \widehat{\boldsymbol{r}}_n, v\boldsymbol{I} + \boldsymbol{Q}^{\mathsf{r}}_n)} \tag{18}$$

$$\widehat{\boldsymbol{x}}_n = C_n^{-1} (\boldsymbol{I} + v^{-1} \boldsymbol{Q}^{\mathsf{r}}_n)^{-1} \widehat{\boldsymbol{r}}_n \tag{19}$$

$$\boldsymbol{Q}^{\mathsf{x}}_n = C_n^{-1} (\boldsymbol{I} + v^{-1} \boldsymbol{Q}^{\mathsf{r}}_n)^{-1} \boldsymbol{Q}^{\mathsf{r}}_n + (C_n - 1) \widehat{\boldsymbol{x}}_n \widehat{\boldsymbol{x}}_n^{\mathsf{T}}, \tag{20}$$

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

2) Inference of $\boldsymbol{z}_m$: When $p_{\mathsf{y}|\mathsf{z}}$ takes the MLR form in (3), closed-form expressions for $\widehat{\boldsymbol{z}}_m(t)$ and $\boldsymbol{Q}^{\mathsf{z}}_m(t)$ from lines 15–16 of Algorithm 1 do not exist. While these computations could be approximated using, e.g., NI or IS, this is expensive because $\widehat{\boldsymbol{z}}_m(t)$ and $\boldsymbol{Q}^{\mathsf{z}}_m(t)$ must be computed for every index m at every HyGAMP iteration t. More details on these approaches will be presented in Section III-C, in the context of SHyGAMP.

D. Implementation of Min-Sum HyGAMP

1) Inference of $\boldsymbol{x}_n$: To ease the computation of line 3 in Algorithm 1, it is typical to choose a log-concave prior p_{x} so that the optimization problem (10) is concave (since $p_{\mathsf{y}|\mathsf{z}}$ in (3) is also log-concave). As discussed in Section I-C, a common example of a log-concave sparsity-promoting prior is the Laplace prior (5). In this case, line 3 becomes

$$\widehat{\boldsymbol{x}}_n = \arg\max_{\boldsymbol{x}} -\frac{1}{2}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}} [\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} (\boldsymbol{x} - \widehat{\boldsymbol{r}}_n) - \lambda \|\boldsymbol{x}\|_1, \tag{21}$$

which is essentially the LASSO [32] problem. Although (21) has no closed-form solution, it can be solved iteratively using, e.g., minorization-maximization (MM) [33].

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

$$\widehat{\boldsymbol{x}}^{(k+1)} = \arg\max_{\boldsymbol{x}} \widehat{J}(\boldsymbol{x}; \widehat{\boldsymbol{x}}^{(k)}), \tag{22}$$

where $\widehat{J}(\boldsymbol{x}; \widehat{\boldsymbol{x}})$ is a surrogate function that minorizes $J(\boldsymbol{x})$ at $\widehat{\boldsymbol{x}}$. In other words, $\widehat{J}(\boldsymbol{x}; \widehat{\boldsymbol{x}}) \leq J(\widehat{\boldsymbol{x}}) \, \forall \boldsymbol{x}$ for any fixed $\widehat{\boldsymbol{x}}$, with equality when $\boldsymbol{x} = \widehat{\boldsymbol{x}}$. To apply MM to (21), we identify the utility function as $J_n(\boldsymbol{x}) \triangleq -\frac{1}{2}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}} [\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} (\boldsymbol{x} - \widehat{\boldsymbol{r}}_n) - \lambda \|\boldsymbol{x}\|_1$. Next we apply a result from [34] that established that $J_n(\boldsymbol{x})$ is minorized by $\widehat{J}_n(\boldsymbol{x}; \widehat{\boldsymbol{x}}^{(k)}_n) \triangleq -\frac{1}{2}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}}$ $[\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} (\boldsymbol{x} - \widehat{\boldsymbol{r}}_n) - \frac{\lambda}{2}\big(\boldsymbol{x}^{\mathsf{T}} \boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}^{(k)}_n) \boldsymbol{x} + \|\widehat{\boldsymbol{x}}^{(k)}_n\|_2^2\big)$ with $\boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}) \triangleq \mathrm{Diag}\big\{|\widehat{x}_1|^{-1}, \dots, |\widehat{x}_D|^{-1}\big\}$. Thus (22) implies

$$\begin{aligned} \widehat{\boldsymbol{x}}^{(k+1)}_n &= \arg\max_{\boldsymbol{x}} \widehat{J}_n(\boldsymbol{x}; \widehat{\boldsymbol{x}}^{(k)}_n) && (23) \\ &= \arg\max_{\boldsymbol{x}} \boldsymbol{x}^{\mathsf{T}} [\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} \widehat{\boldsymbol{r}}_n - \frac{1}{2} \boldsymbol{x}^{\mathsf{T}} \\ &\qquad \big([\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} + \lambda \boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}^{(k)}_n)\big) \boldsymbol{x} && (24) \\ &= \big([\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} + \lambda \boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}^{(k)}_n)\big)^{-1} [\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} \widehat{\boldsymbol{r}}_n && (25) \end{aligned}$$

where (24) dropped the $\boldsymbol{x}$-invariant terms from $\widehat{J}_n(\boldsymbol{x}; \widehat{\boldsymbol{x}}^{(k)}_n)$. Note that each iteration k of (25) requires a $D \times D$ matrix inverse for each n.

Line 4 of Algorithm 1 then says to set $\boldsymbol{Q}^{\mathsf{x}}_n$ equal to the Hessian of the objective function in (21) at $\widehat{\boldsymbol{x}}_n$. Recalling that the second derivative of $|x_{nd}|$ is undefined when $x_{nd} = 0$ but otherwise equals zero, we set $\boldsymbol{Q}^{\mathsf{x}}_n = \boldsymbol{Q}^{\mathsf{r}}_n$ but then zero the dth row and column of $\boldsymbol{Q}^{\mathsf{x}}_n$ for all d such that $\widehat{x}_{nd} = 0$.

2) Inference of $\boldsymbol{z}_m$: Min-sum HyGAMP also requires the computation of lines 12 and 13 in Algorithm 1. In our MLR application, line 12 reduces to the concave optimization problem

$$\widehat{\boldsymbol{z}}_m = \arg\max_{\boldsymbol{z}} -\frac{1}{2}(\boldsymbol{z}-\widehat{\boldsymbol{p}}_m)^{\mathsf{T}}[\boldsymbol{Q}^{\mathsf{p}}_m]^{-1}(\boldsymbol{z}-\widehat{\boldsymbol{p}}_m) + \log p_{\mathsf{y}|\mathbf{z}}(y_m|\boldsymbol{z}). \tag{26}$$

Although (26) can be solved in a variety of ways (see [31] for MM-based methods), we now describe one based on Newton's method [35], i.e.,

$$\widehat{\boldsymbol{z}}^{(k+1)}_m = \widehat{\boldsymbol{z}}^{(k)}_m - \alpha^{(k)}[\boldsymbol{H}^{(k)}_m]^{-1}\boldsymbol{g}^{(k)}_m, \tag{27}$$

where $\boldsymbol{g}^{(k)}_m$ and $\boldsymbol{H}^{(k)}_m$ are the gradient and Hessian of the objective function in (26) at $\widehat{\boldsymbol{z}}^{(k)}_m$, and $\alpha^{(k)} \in (0,1]$ is a stepsize. From (3), it can be seen that $\frac{\partial}{\partial z_i}\log p_{\mathsf{y}|\mathbf{z}}(y|\boldsymbol{z}) = \delta_{y-i} - p_{\mathsf{y}|\mathbf{z}}(i|\boldsymbol{z})$, and so

$$\boldsymbol{g}^{(k)}_m = \boldsymbol{u}(\widehat{\boldsymbol{z}}^{(k)}_m) - \boldsymbol{e}_{y_m} + [\boldsymbol{Q}^{\mathsf{p}}_m]^{-1}(\widehat{\boldsymbol{z}}^{(k)}_m - \widehat{\boldsymbol{p}}_m), \tag{28}$$

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

$$[\boldsymbol{u}(\boldsymbol{z})]_i \triangleq p_{\mathsf{y}|\mathbf{z}}(i|\boldsymbol{z}). \tag{29}$$

Similarly, it is known [36] that the Hessian takes the form

$$\boldsymbol{H}^{(k)}_m = \boldsymbol{u}(\widehat{\boldsymbol{z}}_m)\boldsymbol{u}(\widehat{\boldsymbol{z}}_m)^{\mathsf{T}} - \mathrm{Diag}\{\boldsymbol{u}(\widehat{\boldsymbol{z}}_m)\} - [\boldsymbol{Q}^{\mathsf{p}}_m]^{-1}, \tag{30}$$

which also provides the answer to line 13 of Algorithm 1. Note that each iteration k of (27) requires a $D \times D$ matrix inverse for each m.

It is possible to circumvent the matrix inversion in (27) via componentwise update, i.e.,

$$\widehat{z}^{(k+1)}_{md} = \widehat{z}^{(k)}_{md} - \alpha^{(k)} g^{(k)}_{md}/H^{(k)}_{md}, \tag{31}$$

where $g^{(k)}_{md}$ and $H^{(k)}_{md}$ are the first and second derivatives of the objective function in (26) with respect to z_d at $\boldsymbol{z} = \widehat{\boldsymbol{z}}^{(k)}_m$. From (28)–(30), it follows that

$$g^{(k)}_{md} = p_{\mathsf{y}|\mathbf{z}}(d|\widehat{\boldsymbol{z}}^{(k)}_m) - \delta_{y_m-d} + \big[[\boldsymbol{Q}^{\mathsf{p}}_m]^{-1}\big]^{\mathsf{T}}_{:,d}(\widehat{\boldsymbol{z}}^{(k)}_m - \widehat{\boldsymbol{p}}_m) \tag{32}$$

$$H^{(k)}_{md} = p_{\mathsf{y}|\mathbf{z}}(d|\widehat{\boldsymbol{z}}^{(k)}_m)^2 - p_{\mathsf{y}|\mathbf{z}}(d|\widehat{\boldsymbol{z}}^{(k)}_m) - \big[[\boldsymbol{Q}^{\mathsf{p}}_m]^{-1}\big]_{dd}. \tag{33}$$

E. HyGAMP Summary

In summary, the SPA and MSA variants of the HyGAMP algorithm provide tractable methods of approximating the posterior test-label probabilities $p_{\mathsf{y}_t|\mathbf{y}_{1:M}}\big(y_t \,\big|\, \boldsymbol{y}_{1:M}; \boldsymbol{A}\big)$ and computing the MAP weight matrix $\widehat{\boldsymbol{X}} = \arg\max_{\boldsymbol{X}} p_{\mathbf{y}_{1:M},\mathbf{X}}(\boldsymbol{y}_{1:M}, \boldsymbol{X}; \boldsymbol{A})$, respectively, under a separable likelihood (2) and a separable prior (4). In particular, HyGAMP attacks the high-dimensional inference problems of interest using a sequence of $M+N$ low-dimensional (in particular, D-dimensional) inference problems and $D \times D$ matrix inversions, as detailed in Algorithm 1.

As detailed in the previous sections, however, these D-dimensional inference problems are non-trivial in the sparse MLR case, making HyGAMP computationally costly. We refer the reader to Table I for a summary of the D-dimensional inference problems encountered in running SPA-HyGAMP or MSA-HyGAMP, as well as their associated computational costs. Thus, in the sequel, we propose a computationally efficient simplification of HyGAMP that, as we will see in Section V, compares favorably with existing state-of-the-art methods.

TABLE I
A SUMMARY OF THE D-DIMENSIONAL INFERENCE SUB-PROBLEMS ENCOUNTERED WHEN RUNNING SPA-HYGAMP OR MSA-HYGAMP, AS WELL AS THEIR ASSOCIATED COMPUTATIONAL COSTS

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

'CF' = 'closed form', 'NI' = 'numerical integration', 'MM' = 'minorization-maximization', and 'CWN' = 'component-wise Newton's method'. For the NI method, K denotes the number of samples per dimension, and for the MM and CWN methods K denotes the number of iterations.

III. SHYGAMP FOR MULTICLASS CLASSIFICATION

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

$$\boldsymbol{Q}^{\mathsf{r}}_n = \mathrm{Diag}\left\{q^{\mathsf{r}}_{n1}, \ldots, q^{\mathsf{r}}_{nD}\right\}, \tag{34}$$

and similar for $\boldsymbol{Q}^{\mathsf{x}}_n$, $\boldsymbol{Q}^{\mathsf{p}}_m$, and $\boldsymbol{Q}^{\mathsf{z}}_m$. As a consequence, the $D \times D$ matrix inversions in lines 18 and 20 of Algorithm 1 each reduce to D scalar inversions. More importantly, the D-dimensional inference problems in lines 3–7 and 12–16 can be tackled using much simpler methods than those described in Section II, as we detail below.

A. Scalar Variance Approximation

We further approximate the SHyGAMP algorithm using the *scalar variance* GAMP approximation from [18], which reduces the memory and complexity of the algorithm. The scalar variance approximation first approximates the variances $\{q^{\mathsf{x}}_{nd}\}$ by a value invariant to both n and d, i.e.,

$$q^{\mathsf{x}} \triangleq \frac{1}{ND}\sum_{n=1}^{N}\sum_{d=1}^{D} q^{\mathsf{x}}_{nd}. \tag{35}$$

Then, in line 9 in Algorithm 1, we use the approximation

$$q^{\mathsf{p}}_{md} \approx \sum_{n=1}^{N} A^2_{mn} q^{\mathsf{x}} \overset{(a)}{\approx} \frac{\|\boldsymbol{A}\|^2_F}{M} q^{\mathsf{x}} \triangleq q^{\mathsf{p}}. \tag{36}$$

The approximation (a), after precomputing $\|\boldsymbol{A}\|_F^2$, reduces the complexity of line 9 from $O(ND)$ to $O(1)$. We next define

$$q^{\mathbf{s}} \triangleq \frac{1}{MD} \sum_{m=1}^{M} \sum_{d=1}^{D} q_{md}^{\mathbf{s}} \tag{37}$$

and in line 20 we use the approximation

$$q_{nd}^{\mathbf{r}} \approx \left(\sum_{m=1}^{M} A_{mn}^2 q^{\mathbf{s}} \right)^{-1} \approx \frac{N}{q^{\mathbf{s}} \|\boldsymbol{A}\|_F^2} \triangleq q^{\mathbf{r}}. \tag{38}$$

The complexity of line 20 then simplifies from $O(MD)$ to $O(1)$. For clarity, we note that after applying the scalar variance approximation, we have $\boldsymbol{Q}_n^{\mathbf{x}} = q^{\mathbf{x}} \boldsymbol{I}_D \, \forall n$, and similar for $\boldsymbol{Q}_n^{\mathbf{r}}$, $\boldsymbol{Q}_m^{\mathbf{p}}$ and $\boldsymbol{Q}_m^{\mathbf{z}}$.

B. Sum-Product SHyGAMP: Inference of $\boldsymbol{x}_n$

With diagonal $\boldsymbol{Q}_n^{\mathbf{r}}$ and $\boldsymbol{Q}_n^{\mathbf{x}}$, the implementation of lines 6–7 is greatly simplified by choosing a sparsifying prior $p_{\mathbf{x}}$ with the separable form $p_{\mathbf{x}}(\boldsymbol{x}_n) = \prod_{d=1}^{D} p_{\mathbf{x}}(x_{nd})$. A common example is the Bernoulli-Gaussian (BG) prior

$$p_{\mathbf{x}}(x_{nd}) = (1 - \beta_d)\delta(x_{nd}) + \beta_d \mathcal{N}(x_{nd}; m_d, v_d \boldsymbol{I}). \tag{39}$$

For any separable $p_{\mathbf{x}}$, lines 6 and 7 reduce to computing the mean and variance of the distribution

$$p_{\mathbf{x}|\mathbf{r}}(x_{nd}|\widehat{r}_{nd}; q_{nd}^{\mathbf{r}}) = \frac{p_{\mathbf{x}}(x_{nd}) \mathcal{N}(x_{nd}; \widehat{r}_{nd}, q_{nd}^{\mathbf{r}})}{\int p_{\mathbf{x}}(x'_{nd}) \mathcal{N}(x'_{nd}; \widehat{r}_{nd}, q_{nd}^{\mathbf{r}}) \, \mathrm{d}x'_{nd}}. \tag{40}$$

for all $n = 1 \ldots N$ and $d = 1 \ldots D$, as in the simpler GAMP algorithm [20]. With the BG prior (39), these quantities can be computed in closed form (see, e.g., [37]).

C. Sum-Product SHyGAMP: Inference of $\boldsymbol{z}_m$

With diagonal $\boldsymbol{Q}_m^{\mathbf{p}}$ and $\boldsymbol{Q}_m^{\mathbf{z}}$, the implementation of lines 15 and 16 can also be greatly simplified. Essentially, the problem becomes that of computing the scalar means and variances

$$\widehat{z}_{md} = C_m^{-1} \int_{\mathbb{R}^D} z_d \, p_{\mathbf{y}|\mathbf{z}}(y_m|\boldsymbol{z}) \prod_{k=1}^{D} \mathcal{N}(z_k; \widehat{p}_{mk}, q_{mk}^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{41}$$

$$q_{md}^{\mathbf{z}} = C_m^{-1} \int_{\mathbb{R}^D} z_d^2 \, p_{\mathbf{y}|\mathbf{z}}(y_m|\boldsymbol{z}) \prod_{k=1}^{D} \mathcal{N}(z_k; \widehat{p}_{mk}, q_{mk}^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} - \widehat{z}_{md}^2 \tag{42}$$

for $m = 1 \ldots M$ and $d = 1 \ldots D$. Here, $p_{\mathbf{y}|\mathbf{z}}$ has the MLR form in (3) and C_m is a normalizing constant defined as

$$C_m \triangleq \int_{\mathbb{R}^D} p_{\mathbf{y}|\mathbf{z}}(y_m|\boldsymbol{z}) \prod_{k=1}^{D} \mathcal{N}(z_k; \widehat{p}_{mk}, q_{mk}^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z}. \tag{43}$$

Note that the likelihood $p_{\mathbf{y}|\mathbf{z}}$ is not separable and so inference does not decouple across d, as it did in (40). We now describe several approaches to computing (41) and (42).

1) Numerical Integration: A straightforward approach to (approximately) computing (41)–(43) is through NI. For this, we propose to use a hyper-rectangular grid of $\boldsymbol{z}$ values where, for z_d, the interval $\left[\widehat{p}_{md} - \alpha\sqrt{q_{md}^{\mathbf{p}}}, \widehat{p}_{md} + \alpha\sqrt{q_{md}^{\mathbf{p}}}\right]$ is sampled at K equi-spaced points. Because a D-dimensional numerical integral must be computed for each index m and d, the complexity of this approach grows as $O(MDK^D)$, making it impractical unless D, the number of classes, is very small.

2) Importance Sampling: An alternative approximation of (41)–(43) can be obtained through IS [9, §11.1.4]. Here, we draw K independent samples $\{\widetilde{\boldsymbol{z}}_m[k]\}_{k=1}^K$ from $\mathcal{N}(\widehat{\boldsymbol{p}}_m, \boldsymbol{Q}_m^{\mathbf{p}})$ and compute

$$C_m \approx \sum_{k=1}^{K} p_{\mathbf{y}|\mathbf{z}}(y_m|\widetilde{\boldsymbol{z}}_m[k]) \tag{44}$$

$$\widehat{z}_{md} \approx C_m^{-1} \sum_{k=1}^{K} \widetilde{z}_{md}[k] p_{\mathbf{y}|\mathbf{z}}(y_m|\widetilde{\boldsymbol{z}}_m[k]) \tag{45}$$

$$q_{md}^{\mathbf{z}} \approx C_m^{-1} \sum_{k=1}^{K} \widetilde{z}_{md}^2[k] p_{\mathbf{y}|\mathbf{z}}(y_m|\widetilde{\boldsymbol{z}}_m[k]) - \widehat{z}_{md}^2 \tag{46}$$

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

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

$$\begin{aligned} f_m(\boldsymbol{z}; \widehat{\boldsymbol{p}}_m) &\triangleq p_{\mathbf{y}|\mathbf{z}}(y_m|\widehat{\boldsymbol{p}}_m) + \boldsymbol{g}_m(\widehat{\boldsymbol{p}}_m)^{\mathsf{T}}(\boldsymbol{z} - \widehat{\boldsymbol{p}}_m) \\ &\quad + \frac{1}{2}(\boldsymbol{z} - \widehat{\boldsymbol{p}}_m)^{\mathsf{T}} \boldsymbol{H}_m(\widehat{\boldsymbol{p}}_m)(\boldsymbol{z} - \widehat{\boldsymbol{p}}_m) \end{aligned} \tag{47}$$

for gradient $\boldsymbol{g}_m(\widehat{\boldsymbol{p}}) \triangleq \frac{\partial}{\partial \boldsymbol{z}} p_{\mathbf{y}|\mathbf{z}}(y_m|\boldsymbol{z})\big|_{\boldsymbol{z}=\widehat{\boldsymbol{p}}}$ and Hessian $\boldsymbol{H}_m(\widehat{\boldsymbol{p}}) \triangleq \frac{\partial^2}{\partial \boldsymbol{z}^2} p_{\mathbf{y}|\mathbf{z}}(y_m|\boldsymbol{z})\big|_{\boldsymbol{z}=\widehat{\boldsymbol{p}}}$. In this case, it can be shown [31] that

$$C_m \approx f_m(\widehat{\boldsymbol{p}}_m) + \frac{1}{2} \sum_{k=1}^{D} H_{mk}(\widehat{\boldsymbol{p}}_m) q_{mk}^{\mathbf{p}} \tag{48}$$

$$\begin{aligned} \widehat{z}_{md} \approx \widehat{C}_m^{-1} \Bigg(& f_m(\widehat{\boldsymbol{p}}_m) \widehat{p}_{md} + g_{md}(\widehat{\boldsymbol{p}}_m) q_{md}^{\mathbf{p}} \\ & + \frac{1}{2} \sum_{k=1}^{D} \widehat{p}_{mk} q_{mk}^{\mathbf{p}} H_{mk}(\widehat{\boldsymbol{p}}_m) \Bigg) \end{aligned} \tag{49}$$

$$\begin{aligned} q_{md}^{\mathbf{z}} \approx C_m^{-1} \Bigg(& f_m(\widehat{\boldsymbol{p}}_m) (\widehat{p}_{md}^2 + q_{md}^{\mathbf{p}}) + 2 g_{md}(\widehat{\boldsymbol{p}}_m) \widehat{p}_{md} q_{md}^{\mathbf{p}} \\ & + \frac{1}{2} q_{md}^{\mathbf{p}} (\widehat{p}_{md}^2 + 3 q_{md}^{\mathbf{p}}) H_{md}(\widehat{\boldsymbol{p}}_m) \\ & + \frac{1}{2} (\widehat{p}_{md}^2 + q_{md}^{\mathbf{p}}) H_{md}(\widehat{\boldsymbol{p}}_m) \sum_{k \neq d} q_{mk}^{\mathbf{p}} \Bigg) - \widehat{z}_{md}^2, \end{aligned} \tag{50}$$

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

4) Gaussian Mixture Approximation: It is known that the logistic cdf $1/(1 + \exp(-x))$ is well approximated by a mixture of a few Gaussian cdfs, which leads to an efficient method of approximating (41) and (42) in the case of *binary* logistic regression (i.e., $D = 2$) [38]. We now develop an extension of this method for the MLR case (i.e., $D \geq 2$).

To facilitate the GM approximation, we work with the difference variables

$$\gamma_d^{(y)} \triangleq \begin{cases} z_y - z_d & d \neq y \\ z_y & d = y. \end{cases} \tag{51}$$

Their utility can be seen from the fact that (recalling (3))

$$p_{\mathsf{y}|\mathbf{z}}(y|\boldsymbol{z}) = \frac{1}{1 + \sum_{d \neq y} \exp(z_d - z_y)} \tag{52}$$

$$= \frac{1}{1 + \sum_{d \neq y} \exp\left(-\gamma_d^{(y)}\right)} \triangleq l^{(y)}\left(\boldsymbol{\gamma}^{(y)}\right), \tag{53}$$

which is smooth, positive, and bounded by 1, and strictly increasing in $\gamma_d^{(y)}$. Thus,[4] for appropriately chosen $\{\alpha_l, \mu_{kl}, \sigma_{kl}\}$,

$$l^{(y)}(\boldsymbol{\gamma}) \approx \sum_{l=1}^{L} \alpha_l \prod_{k \neq y} \Phi\left(\frac{\gamma_k - \mu_{kl}}{\sigma_{kl}}\right) \triangleq \widehat{l}^{(y)}(\boldsymbol{\gamma}), \tag{54}$$

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

Recall from (41)–(43) that our objective is to compute quantities of the form

$$\int_{\mathbb{R}^D} \left(\boldsymbol{e}_d^{\mathsf{T}} \boldsymbol{z}\right)^i p_{\mathsf{y}|\mathbf{z}}(y|\boldsymbol{z}) \mathcal{N}(\boldsymbol{z}; \widehat{\boldsymbol{p}}, \boldsymbol{Q}^{\mathsf{p}}) \, \mathrm{d}\boldsymbol{z} \triangleq S_{di}^{(y)}, \tag{55}$$

where $i \in \{0, 1, 2\}$, $\boldsymbol{Q}^{\mathsf{p}}$ is diagonal, and $\boldsymbol{e}_d$ is the dth column of $\boldsymbol{I}_D$. To exploit (54), we change the integration variable to

$$\boldsymbol{\gamma}^{(y)} = \boldsymbol{T}_y \boldsymbol{z} \tag{56}$$

with

$$\boldsymbol{T}_y = \begin{bmatrix} -\boldsymbol{I}_{y-1} & \mathbf{1}_{(y-1)\times 1} & \mathbf{0}_{(y-1)\times(D-y)} \\ \mathbf{0}_{1\times(y-1)} & 1 & \mathbf{0}_{1\times(D-y)} \\ \mathbf{0}_{(D-y)\times(y-1)} & \mathbf{1}_{(D-y)\times 1} & -\boldsymbol{I}_{D-y} \end{bmatrix} \tag{57}$$

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

$$S_{di}^{(y)} = \int_{\mathbb{R}^D} \left(\boldsymbol{e}_d^{\mathsf{T}} \boldsymbol{T}_y^{-1} \boldsymbol{\gamma}\right)^i l^{(y)}(\boldsymbol{\gamma}) \mathcal{N}\left(\boldsymbol{\gamma}; \boldsymbol{T}_y \widehat{\boldsymbol{p}}, \boldsymbol{T}_y \boldsymbol{Q}^{\mathsf{p}} \boldsymbol{T}_y^{\mathsf{T}}\right) \mathrm{d}\boldsymbol{\gamma}. \tag{58}$$

Then, applying the approximation (54) and

$$\mathcal{N}\left(\boldsymbol{\gamma}; \boldsymbol{T}_y \widehat{\boldsymbol{p}}, \boldsymbol{T}_y \boldsymbol{Q}^{\mathsf{p}} \boldsymbol{T}_y^{\mathsf{T}}\right) = \mathcal{N}\left(\gamma_y; \widehat{p}_y, q_y^{\mathsf{p}}\right) \times \prod_{k \neq y} \mathcal{N}\left(\gamma_k; \gamma_y - \widehat{p}_k, q_k^{\mathsf{p}}\right) \tag{59}$$

to (58), we find that

$$S_{di}^{(y)} \approx \sum_{l=1}^{L} \alpha_l \int_{\mathbb{R}} \mathcal{N}\left(\gamma_y; \widehat{p}_y, q_y^{\mathsf{p}}\right) \Bigg[\int_{\mathbb{R}^{D-1}} \left(\boldsymbol{e}_d^{\mathsf{T}} \boldsymbol{T}_y^{-1} \boldsymbol{\gamma}\right)^i \times \prod_{k \neq y} \mathcal{N}\left(\gamma_k; \gamma_y - \widehat{p}_k, q_k^{\mathsf{p}}\right) \Phi\left(\frac{\gamma_k - \mu_{kl}}{\sigma_{kl}}\right) \mathrm{d}\gamma_k \Bigg] \mathrm{d}\gamma_y. \tag{60}$$

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

$$\boldsymbol{e}_d^{\mathsf{T}} \boldsymbol{T}_y^{-1} \boldsymbol{\gamma} = \begin{cases} \gamma_y - \gamma_d & d \neq y \\ \gamma_y & d = y. \end{cases} \tag{61}$$

Thus, for a fixed value of $\gamma_y = c$, the inner integral in (60) can be expressed as a product of linear combinations of terms

$$\int_{\mathbb{R}} \gamma^i \mathcal{N}(\gamma; c - \widehat{p}, q) \Phi\left(\frac{\gamma - \mu}{\sigma}\right) \mathrm{d}\gamma \triangleq T_i \tag{62}$$

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

$$T_0 = \Phi(x) \tag{63}$$

$$T_1 = (c - \widehat{p})\Phi(x) + \frac{q\phi(x)}{\sqrt{\sigma^2 + q}} \tag{64}$$

$$T_2 = \frac{(T_1)^2}{\Phi(x)} + q\Phi(x) - \frac{q^2 \phi(x)}{\sigma^2 + q}\left(x + \frac{\phi(x)}{\Phi(x)}\right), \tag{65}$$

which can be obtained using the results in [39, §3.9]. The outer integral in (60) can then be approximated via NI.

If a grid of K values is used for NI over γ_y in (60), then the overall complexity of the method grows as $O(MDLK)$. Our experiments indicate that relatively small values (e.g., $L = 2$ and $K = 7$) suffice.

5) Performance Comparison: Above we described four methods of approximating lines 15 and 16 in Algorithm 1 under diagonal $\boldsymbol{Q}^{\mathsf{p}}$ and $\boldsymbol{Q}^{\mathsf{z}}$. We now compare the accuracy and complexity of these methods. In particular, we measured the accuracy of the conditional mean (i.e., line 15) approximation as follows (for a given $\widehat{\boldsymbol{p}}$ and $\boldsymbol{Q}^{\mathsf{p}}$):

1) generate i.i.d. samples $\boldsymbol{z}_{\mathsf{true}}[t] \sim \mathcal{N}(\boldsymbol{z}; \widehat{\boldsymbol{p}}, \boldsymbol{Q}^{\mathsf{p}})$ and $y_{\mathsf{true}}[t] \sim p_{\mathsf{y}|\mathbf{z}}(y \,|\, \boldsymbol{z}_{\mathsf{true}}[t])$ for $t = 1 \ldots T$,
2) compute the approximation $\widehat{\boldsymbol{z}}[t] \approx \mathrm{E}\{\mathbf{z} \,|\, \mathsf{y} = y_{\mathsf{true}}[t], \mathbf{p} = \widehat{\boldsymbol{p}}; \boldsymbol{Q}^{\mathsf{p}}\}$ using each method described in Sections III-C1–III-C4,
3) compute average MSE $\triangleq \frac{1}{T} \sum_{t=1}^{T} \left\| \boldsymbol{z}_{\mathsf{true}}[t] - \widehat{\boldsymbol{z}}[t] \right\|_2^2$ for each method,

and we measured the combined runtime of lines 15 and 16 for each method. Unless otherwise noted, we used $D = 4$ classes, $\widehat{\boldsymbol{p}} = \boldsymbol{e}_1$, $\boldsymbol{Q}^{\mathsf{p}} = q^{\mathsf{p}} \boldsymbol{I}_D$, and $q^{\mathsf{p}} = 1$ in our experiments. For NI, we used a grid of size $K = 7$ and radius of $\alpha = 4$ standard deviations; for IS, we used $K = 1500$ samples; and for the GM method, we used $L = 2$ mixture components and a grid size of $K = 7$. Empirically, we found that smaller grids or fewer samples compromised accuracy, whereas larger grids or more samples compromised runtime.

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

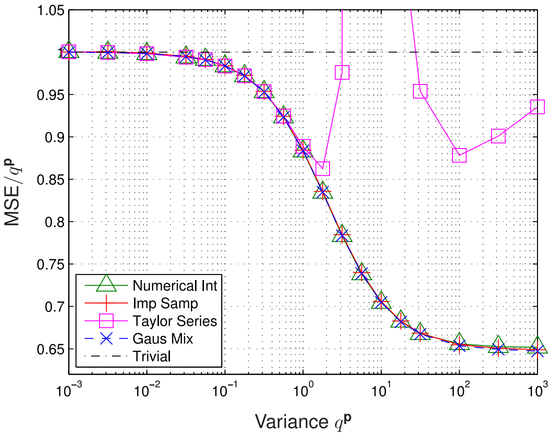


Fig. 2. MSE/q^{p} versus variance q^{p} for various methods to compute line 15 in Algorithm 1. Each point represents the average of 5×10^6 independent trials.

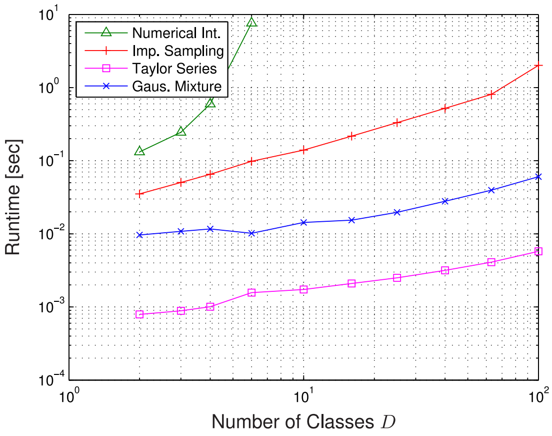


Fig. 3. Cumulative runtime (over $M = 500$ samples) versus number-of-classes D for various methods to compute lines 15 and 16 in Algorithm 1. Each point represents the average of 2000 independent trials.

Fig. 2 plots the normalized MSE versus variance q^{p} for the four methods under test, in addition to the trivial method $\widehat{z}[t] = \widehat{\boldsymbol{p}}$. The figure shows that the NI, IS, and GM methods performed similarly across the full range of q^{p} and always outperform the trivial method. The TS method, however, breaks down when $q^{\mathsf{p}} > 1$. A close examination of the figure reveals that GM gave the best accuracy, IS the second best accuracy, and NI the third best accuracy.

Fig. 3 shows the cumulative runtime (over $M = 500$ training samples) of the methods from Sections III-C1–III-C4 versus the number of classes, D. Although the TS method was the fastest, we saw in Fig. 2 that it is accurate only at small variances q^{p}. Fig. 3 then shows GM was about an order-of-magnitude faster than IS, which was several orders-of-magnitude faster than NI.

Together, Figs. 2 and 3, show that our proposed GM method dominated the IS and NI methods in both accuracy and runtime. Thus, for the remainder of the paper, we implement sum-product SHyGAMP using the GM method from Section III-C4.

D. Min-Sum SHyGAMP: Inference of $\boldsymbol{x}_n$

With diagonal $\boldsymbol{Q}^{\mathsf{r}}_n$ and $\boldsymbol{Q}^{\mathsf{x}}_n$, the implementation of lines 3 and 4 in Algorithm 1 can be significantly simplified. Recall that, when the prior p_{x} is chosen as i.i.d. Laplace (5), line 3 manifests as (21), which is in general a non-trivial optimization problem. But with diagonal $\boldsymbol{Q}^{\mathsf{r}}_n$, (21) decouples into D instances of the scalar optimization

$$x_{nd} = \arg\max_{x} -\frac{1}{2}\frac{(x - \widehat{r}_{nd})^2}{q^{\mathsf{r}}_{nd}} - \lambda|x|, \tag{66}$$

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

$$\widehat{x}_{nd} = \operatorname{sgn}(\widehat{r}_{nd}) \max\{0, |\widehat{r}_{nd}| - \lambda q^{\mathsf{r}}_{nd}\}. \tag{67}$$

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

Meanwhile, line 4 reduces to

$$q^{\mathsf{x}}_{nd} = \left[\frac{\partial^2}{\partial x^2} \left(\frac{1}{2}\frac{(x - \widehat{r}_{nd})^2}{q^{\mathsf{r}}_{nd}} + \lambda|x| \right)\bigg|_{x=\widehat{x}_{nd}} \right]^{-1}, \tag{68}$$

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

E. Min-Sum SHyGAMP: Inference of $\boldsymbol{z}_m$

With diagonal $\boldsymbol{Q}^{\mathsf{p}}_m$ and $\boldsymbol{Q}^{\mathsf{z}}_m$, the implementation of lines 12 and 13 in Algorithm 1 also simplifies. Recall that, when the likelihood $p_{\mathsf{y}|\mathbf{z}}$ takes the MLR form in (3), line 12 manifests as (26), which can be solved using a component-wise Newton's method as in (31)–(33) for any $\boldsymbol{Q}^{\mathsf{p}}_m$ and $\boldsymbol{Q}^{\mathsf{z}}_m$. When $\boldsymbol{Q}^{\mathsf{p}}_m$ is diagonal, the first and second derivatives (32) and (33) reduce to

$$g^{(k)}_{md} = p_{\mathsf{y}|\mathbf{z}}\Big(d|\widehat{\boldsymbol{z}}^{(k)}_m\Big) - \delta_{y_m - d} + \Big(\widehat{z}^{(k)}_{md} - \widehat{p}_{md}\Big)/q^{\mathsf{p}}_{md} \tag{69}$$

$$H^{(k)}_{md} = p_{\mathsf{y}|\mathbf{z}}\Big(d|\widehat{\boldsymbol{z}}^{(k)}_m\Big)^2 - p_{\mathsf{y}|\mathbf{z}}\Big(d|\widehat{\boldsymbol{z}}^{(k)}_m\Big) - 1/q^{\mathsf{p}}_{md}, \tag{70}$$

which leads to a reduction in complexity.

Furthermore, line 13 simplifies, since with diagonal $\boldsymbol{Q}^{\mathsf{z}}_m$ it suffices to compute only the diagonal components of $\boldsymbol{H}^{(k)}_m$ in (30). In particular, when $\boldsymbol{Q}^{\mathsf{p}}_m$ is diagonal, the result becomes

$$q^{\mathsf{z}}_{md} = \frac{1}{1/q^{\mathsf{p}}_{md} + p_{\mathsf{y}|\mathbf{z}}(d|\widehat{\boldsymbol{z}}_m) - p_{\mathsf{y}|\mathbf{z}}(d|\widehat{\boldsymbol{z}}_m)^2}. \tag{71}$$

F. SHyGAMP Summary

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

IV. Online Parameter Tuning

The weight vector priors in (5) and (39) depend on modeling parameters that, in practice, must be tuned. Although CV is the customary approach to tuning such model parameters, it can be very computationally costly, since each parameter must be tested over a grid of hypothesized values and over multiple data folds. For example, K-fold CV tuning of P parameters using G

TABLE II
A SUMMARY OF THE D-DIMENSIONAL INFERENCE SUB-PROBLEMS ENCOUNTERED WHEN RUNNING SPA-SHYGAMP OR MSA-SHYGAMP, AS WELL AS THEIR ASSOCIATED COMPUTATIONAL COSTS

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

'CF' = 'closed form', 'GM' = 'Gaussian mixture', 'ST' = 'Soft-thresholding', and 'CWN' = 'component-wise Newton's method'. For the GM, L denotes the number of mixture components and K the number of samples in the 1D numerical integral, and for CWN K denotes the number of iterations.

TABLE III
HIGH-LEVEL COMPARISON OF SHYGAMP AND HYGAMP

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

hypothesized values of each parameter requires the training and evaluation of KG^P classifiers.

A. Parameter Selection for Sum-Product SHyGAMP

For SPA-SHyGAMP, we propose to use the zero-mean BG prior in (39), which has parameters β_d, m_d, and v_d. Instead of CV, we use the EM-GM-AMP framework described in [25] to tune these parameters online. See [31] for details regarding the initialization of β_d, m_d, and v_d.

B. Parameter Selection for Min-Sum SHyGAMP

To use MSA-SHyGAMP with the Laplacian prior in (5), we need to specify the scale parameter λ. For this, we use a modification of the SURE-AMP framework from [26], which adjusts λ to minimize the SURE of the weight-vector MSE.

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

$$\mathsf{r} = \mathsf{x} + \sqrt{q^{\mathsf{r}}}\mathsf{w}, \tag{72}$$

with $\mathsf{w} \sim \mathcal{N}(0,1)$ and $q^{\mathsf{r}} > 0$. For this purpose, consider an estimate of the form $\widehat{x} = f(r, q^{\mathsf{r}}; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ contains tunable parameters. For convenience, define the shifted estimation function $g(r, q^{\mathsf{r}}; \boldsymbol{\theta}) \triangleq f(r, q^{\mathsf{r}}; \boldsymbol{\theta}) - r$ and its derivative $g'(r, q^{\mathsf{r}}; \boldsymbol{\theta}) \triangleq \frac{\partial}{\partial r} g(r, q^{\mathsf{r}}; \boldsymbol{\theta})$. Then Stein [40] established the following result on the mean-squared error, or risk, of the estimate $\widehat{x}$:

$$\mathrm{E}\left\{[\widehat{\mathsf{x}} - \mathsf{x}]^2\right\} = q^{\mathsf{r}} + \mathrm{E}\left\{g^2(\mathsf{r}, q^{\mathsf{r}}; \boldsymbol{\theta}) + 2q^{\mathsf{r}} g'(\mathsf{r}, q^{\mathsf{r}}; \boldsymbol{\theta})\right\}. \tag{73}$$

The implication of (73) is that, given only the noisy observation r and the noise variance q^{r}, one can compute an estimate

$$\mathrm{SURE}(r, q^{\mathsf{r}}; \boldsymbol{\theta}) \triangleq q^{\mathsf{r}} + g^2\left(r, q^{\mathsf{r}}; \boldsymbol{\theta}\right) + 2q^{\mathsf{r}} g'(r, q^{\mathsf{r}}; \boldsymbol{\theta}) \tag{74}$$

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

$$\mathrm{E}\left\{\mathrm{SURE}(\mathsf{r}, q^{\mathsf{r}}; \boldsymbol{\theta})\right\} = \mathrm{MSE}(\boldsymbol{\theta}). \tag{75}$$

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

In [26], it was noticed that the assumption (72) is satisfied by AMP's denoiser inputs $\{\widehat{r}_n\}_{n=1}^N$, and thus [26] proposed to tune the soft threshold λ to minimize the SURE:

$$\widehat{\lambda} = \arg\min_{\lambda} \sum_{n=1}^{N} g^2\left(\widehat{r}_n, q^{\mathsf{r}}; \lambda\right) + 2q^{\mathsf{r}} g'(\widehat{r}_n, q^{\mathsf{r}}; \lambda). \tag{76}$$

Recalling the form of the estimator $f(\cdot)$ from (67), we have

$$g^2(\widehat{r}_n, q^{\mathsf{r}}; \lambda) = \begin{cases} \lambda^2 (q^{\mathsf{r}})^2 & \text{if } |\widehat{r}_n| > \lambda q^{\mathsf{r}} \\ \widehat{r}_n^2 & \text{otherwise} \end{cases} \tag{77}$$

$$g'(\widehat{r}_n, q^{\mathsf{r}}; \lambda) = \begin{cases} -1 & \text{if } |\widehat{r}_n| < \lambda q^{\mathsf{r}} \\ 0 & \text{otherwise.} \end{cases} \tag{78}$$

However, solving (76) for λ is non-trivial because the objective is non-smooth and has many local minima. A stochastic gradient descent approach was proposed in [26], but its convergence speed is too slow to be practical.

Since (72) also matches the scalar-variance SHyGAMP model from Section III-A, we propose to use SURE to tune λ for min-sum SHyGAMP. But, instead of the empirical average in (76), we propose to use a statistical average, i.e.,

$$\widehat{\lambda} = \arg\min_{\lambda} \underbrace{\mathrm{E}\left\{g^2\left(\mathsf{r}, q^{\mathsf{r}}; \lambda\right) + 2q^{\mathsf{r}} g'(\mathsf{r}, q^{\mathsf{r}}; \lambda)\right\}}_{\triangleq J(\lambda)}, \tag{79}$$

by modeling the random variable r as a GM whose parameters are fitted to $\{\widehat{r}_{nd}\}$. As a result, the objective in (79) is smooth. Moreover, by constraining the smallest mixture variance to be at least q^{r}, the objective becomes unimodal, in which case $\widehat{\lambda}$ from (79) is the unique root of $\frac{\mathrm{d}}{\mathrm{d}\lambda} J(\lambda)$. To find this root, we use the bisection method. In particular, due to (77)–(78), the objective in (79) becomes

$$J(\lambda) = \int_{-\infty}^{-\lambda q^{\mathsf{r}}} p_{\mathsf{r}}(r) \lambda^2 (q^{\mathsf{r}})^2 \,\mathrm{d}r + \int_{-\lambda q^{\mathsf{r}}}^{\lambda q^{\mathsf{r}}} p_{\mathsf{r}}(r)(r^2 - 2q^{\mathsf{r}}) \,\mathrm{d}r + \int_{\lambda q^{\mathsf{r}}}^{\infty} p_{\mathsf{r}}(r) \lambda^2 (q^{\mathsf{r}})^2 \,\mathrm{d}r, \tag{80}$$

from which it can be shown that [31]

$$\frac{\mathrm{d}}{\mathrm{d}\lambda} J(\lambda) = 2\lambda (q^{\mathsf{r}})^2 \left[1 - \Pr\{-\lambda q^{\mathsf{r}} < \mathsf{r} < \lambda q^{\mathsf{r}}\}\right] - \left[p_{\mathsf{r}}(\lambda q^{\mathsf{r}}) + p_{\mathsf{r}}(-\lambda q^{\mathsf{r}})\right] 2(q^{\mathsf{r}})^2. \tag{81}$$

For GM fitting, we use the standard EM approach [9] and find that relatively few (e.g., $L = 3$) mixture terms suffice. Note that we re-tune λ using the above technique at each iteration of Algorithm 1, immediately before line 3. Experimental verification of our method is provided in Section V-B.

V. Numerical Results

In this section we describe the results of several experiments used to test SHyGAMP. In these experiments, EM-tuned SPA-SHyGAMP and SURE-tuned MSA-SHyGAMP were compared to two state-of-the-art sparse MLR algorithms: SBMLR [14] and GLMNET [13]. We are particularly interested in SBMLR and GLMNET because [13], [14] show that they have strong advantages over earlier algorithms, e.g., [10]–[12]. As described in Section I-C, both SBMLR and GLMNET use ℓ_1 regularization, but SBMLR tunes the regularization parameter λ using evidence maximization while GLMNET tunes it using CV (using the default value of 10 folds unless otherwise noted). For SBMLR and GLMNET, we ran code written by the authors[5,6] under default settings (unless otherwise noted). For SHyGAMP, we used the damping modification described in [30]. We note that the runtimes reported for all algorithms include the total time spent to tune all parameters and train the final classifier.

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

A. Synthetic Data in the $M \ll N$ Regime

We first describe the results of three experiments with synthetic data. For these experiments, the training data were randomly generated and algorithm performance was averaged over several data realizations. In all cases, we started with balanced training labels $y_m \in \{1,\ldots,D\}$ for $m = 1,\ldots,M$ (i.e., M/D examples from each of D classes). Then, for each data realization, we generated M i.i.d. training features $\boldsymbol{a}_m$ from the class-conditional generative distribution $\boldsymbol{a}_m \,|\, y_m \sim \mathcal{N}(\boldsymbol{\mu}_{y_m}, v\boldsymbol{I}_N)$. In doing so, we chose the intra-class variance, v, to attain a desired Bayes error rate (BER) of 10% (see [31] for details), and we used randomly generated K-sparse orthonormal class means, $\boldsymbol{\mu}_d \in \mathbb{R}^N$. In particular, we generated $[\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_D]$ by drawing a $K \times K$ matrix with i.i.d. $\mathcal{N}(0,1)$ entries, performing a singular value decomposition, and zero-padding the first D left singular vectors to length N. We note that our generation of $\boldsymbol{y}, \boldsymbol{A}, \boldsymbol{X}$ is matched [41] to the multinomial logistic models (2) and (3).

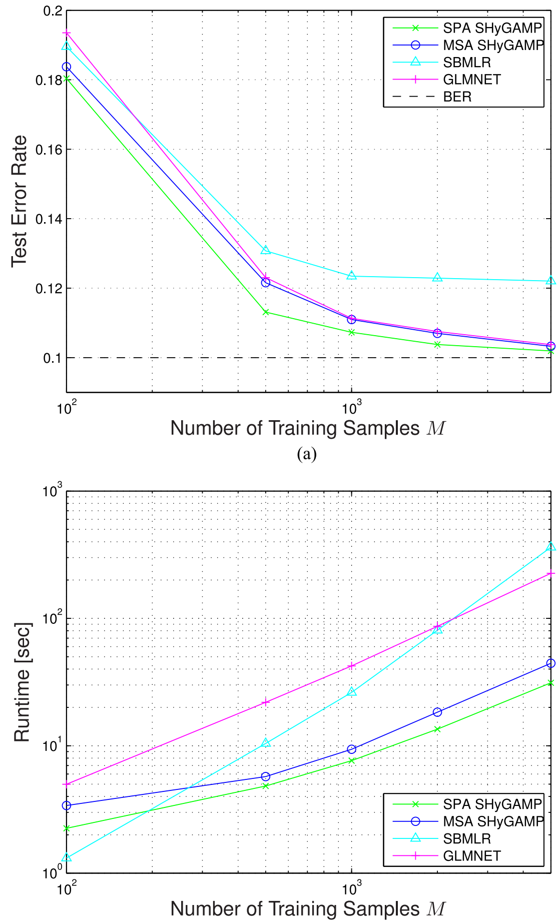
Given a training data realization, each algorithm was invoked to yield a weight matrix $\widehat{\boldsymbol{X}} = [\widehat{\boldsymbol{x}}_1,\ldots,\widehat{\boldsymbol{x}}_D]$. The corresponding expected test-error rate was then analytically computed as

$$\Pr\{\text{err}\} = 1 - \frac{1}{D}\sum_{y=1}^{D} \Pr\{\text{cor}|y\} \tag{82}$$

$$\Pr\{\text{cor}|y\} = \Pr \bigcap_{d \neq y} \left\{ (\widehat{\boldsymbol{x}}_y - \widehat{\boldsymbol{x}}_d)^{\mathsf{T}} \mathbf{a} < (\widehat{\boldsymbol{x}}_y - \widehat{\boldsymbol{x}}_d)^{\mathsf{T}} \boldsymbol{\mu}_y \right\}, \tag{83}$$

TABLE IV
Configurations of the Synthetic-Data Experiments

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



(a)

(b)

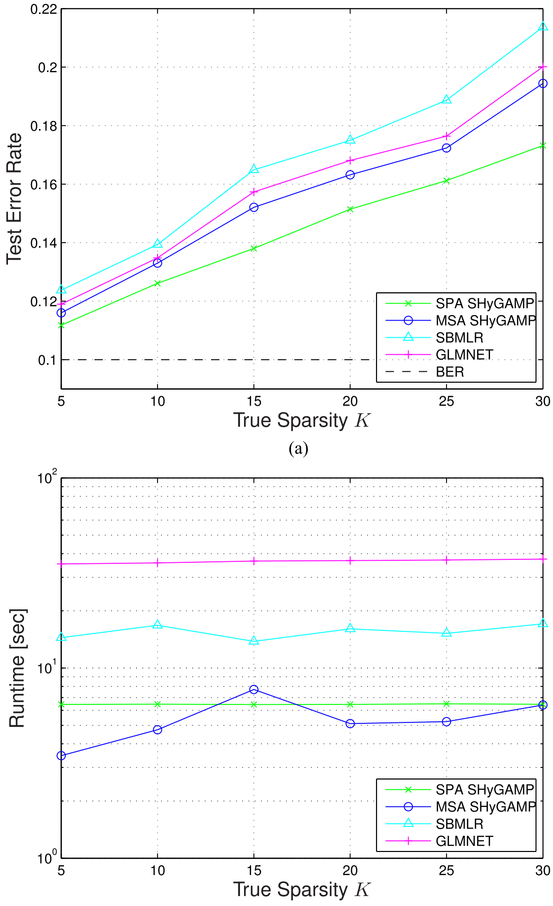
Fig. 4. Synthetic Experiment 1: expected test-error rate and runtime versus M. Here, $D = 4$, $N = 10\,000$, and $K = 10$. (a) Error. (b) Runtime.

where $\mathbf{a} \sim \mathcal{N}(\boldsymbol{0}, v\boldsymbol{I}_N)$ and the multivariate normal cdf in (83) was computed using Matlab's `mvncdf`.

For all three synthetic-data experiments, we used $D = 4$ classes and $K \ll M \ll N$. In the first experiment, we fixed K and N and we varied M; in the second experiment, we fixed K and M and we varied K; and in the third experiment, we fixed K and M and we varied N. The specific values/ranges of K, M, N used for each experiment are given in Table IV.

Fig. 4(a) and (b) show the expected test-error rate and runtime, respectively, versus the number of training examples, M, averaged over 12 independent trials. Fig. 4(a) shows that, at all tested values of M, SPA-SHyGAMP gave the best error-rates and MSA-SHyGAMP gave the second best error-rates, although those reached by GLMNET were similar at large M. Moreover, the error-rates of SPA-SHyGAMP, MSA-SHyGAMP, and

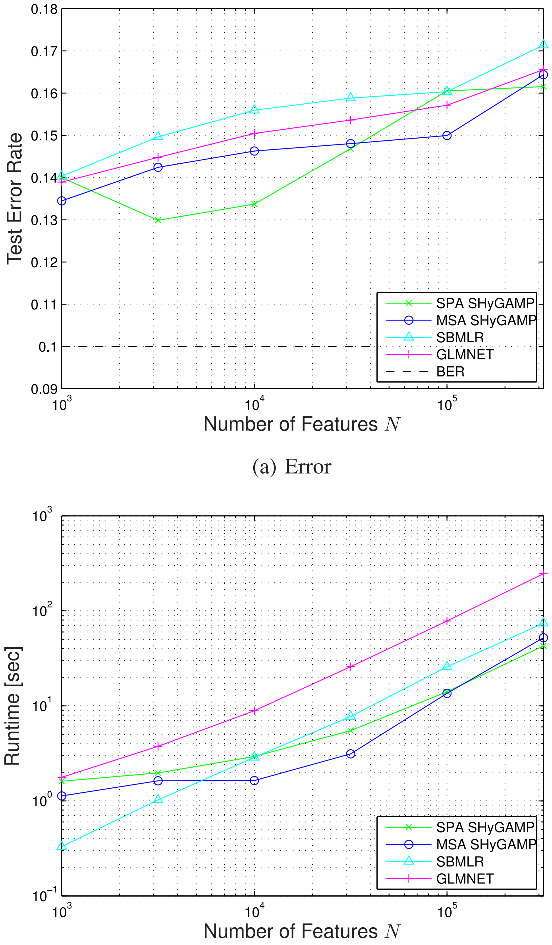
[5] SBMLR obtained from http://theoval.cmp.uea.ac.uk/matlab/
[6] GLMNET obtained from http://www.stanford.edu/~hastie/glmnet_matlab/



(a)

(b)

Fig. 5. Synthetic Experiment 2: expected test-error rate and runtime versus K. Here, $D = 4$, $M = 300$, and $N = 30\,000$. (a) Error. (b) Runtime.



(a) Error

(b) Runtime

Fig. 6. Synthetic Experiment 3: expected test-error rate and runtime versus N. Here, $D = 4$, $M = 200$, and $K = 10$. (a) Error. (b) Runtime.

GLMNET all converged towards the BER as M increased, whereas that of SBMLR did not. Since MSA-SHyGAMP, GLMNET, and SBMLR all solve the same ℓ_1-regularized MLR problem, the difference in their error-rates can be attributed to the difference in their tuning of the regularization parameter λ. Fig. 4(b) shows that, for $M > 500$, SPA-SHyGAMP was the fastest, followed by MSA-SHyGAMP, SBMLR, and GLMNET. Note that the runtimes of SPA-SHyGAMP, MSA-SHyGAMP, and GLMNET increased linearly with M, whereas the runtime of SBMLR increased quadratically with M.

Fig. 5(a) and (b) show the expected test-error rate and runtime, respectively, versus feature-vector sparsity, K, averaged over 12 independent trials. Fig. 5(a) shows that, at all tested values of K, SPA-SHyGAMP gave the best error-rates and MSA-SHyGAMP gave the second best error-rates. Fig. 5(b) shows that SPA-SHyGAMP and MSA-SHyGAMP gave the fastest runtimes. All runtimes were approximately invariant to K.

Fig. 6(a) and (b) show the expected test-error rate and runtime, respectively, versus the number of features, N, averaged over 12 independent trials. Fig. 6(a) shows that, at all tested values of N, MSA-SHyGAMP gave lower error-rates than SBMLR and GLMNET. Meanwhile, SPA-SHyGAMP gave the lowest error-rates for certain values of N. Fig. 6(b) shows that SPA-SHyGAMP and MSA-SHyGAMP gave the fastest runtimes for $N \geq 10\,000$, while SBMLR gave the fastest runtimes fo $N \leq 3000$. All runtimes increased linearly with N.

B. Example of SURE Tuning

Although the good error-rate performance of MSA-SHyGAMP in Section V-A suggests that the SURE λ-tuning method from Section IV-B is working reliably, we now describe a more direct test of its behavior. Using synthetic data generated as described in Section V-A with $D = 4$ classes, $N = 30\,000$ features, $M = 300$ examples, and sparsity $K = 25$, we ran MSA-SHyGAMP using various fixed values of λ. In the sequel, we refer to this experiment as "Synthetic Experiment 4." The resulting expected test-error rate versus λ (averaged over 10 independent realizations) is shown in Fig. 7. For the same realizations, we ran MSE-SHyGAMP with SURE-tuning and plot the resulting error-rate and average $\widehat{\lambda}$ in Fig. 7. From Fig. 7, we see that the SURE λ-tuning method matched both the minimizer and the minimum of the error-versus-λ trace of fixed-λ MSA-SHyGAMP.

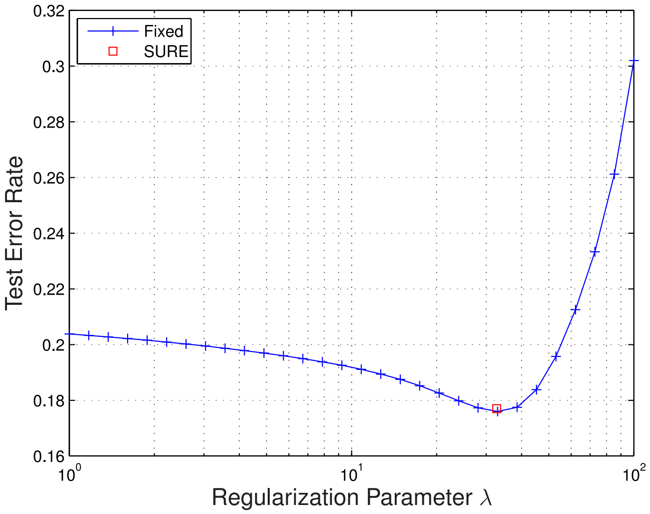


Fig. 7. Synthetic experiment 4: expected test-error rate versus regularization parameter λ for fixed-λ MSA-SHyGAMP. Here, $D = 4$, $M = 300$, $N = 30\,000$, and $K = 25$. Also shown is the average test-error rate for SURE-tuned MSA-SHyGAMP plotted at the average value of $\widehat{\lambda}$.

TABLE V
ESTIMATED TEST-ERROR RATE, STANDARD DEVIATION OF ESTIMATE, RUNTIME, AND SPARSITIES FOR THE SUN DATASET

Algorithm	% Error (SD)	Runtime (s)	$\widehat{K}_{99}$	$\Vert\widehat{\boldsymbol{X}}\Vert_0$
SPA-SHyGAMP	33.3 (3.8)	6.86	20.05	218 452
MSA-SHyGAMP	31.0 (3.7)	13.59	93.00	145.32
SBMLR	31.6 (3.7)	22.48	49.89	72.89
GLMNET	33.9 (3.8)	31.93	10.89	16.84

TABLE VI
ESTIMATED TEST-ERROR RATE, STANDARD DEVIATION OF ESTIMATE, RUNTIME, AND SPARSITIES FOR THE BHATTACHARJEE DATASET

Algorithm	% Error (SD)	Runtime (s)	$\widehat{K}_{99}$	$\Vert\widehat{\boldsymbol{X}}\Vert_0$
SPA-SHyGAMP	9.5 (2.1)	3.26	16.15	63 000
MSA-SHyGAMP	10.5 (2.2)	6.11	55.20	84.65
SBMLR	9.5 (2.1)	6.65	44.25	79.10
GLMNET	12.0 (2.4)	13.67	49.65	89.40

C. Micro-Array Gene Expression

Next we consider classification and feature-selection using micro-array gene expression data. Here, the labels indicate which type of disease is present (or no disease) and the features represent gene expression levels. The objective is 1) to determine which subset of genes best predicts the various diseases and 2) to classify whether an (undiagnosed) patient is at risk for any of these diseases based on their gene profile.

We tried two datasets: one from Sun *et al.* [1] and one from Bhattacharjee *et al.* [2]. The Sun dataset includes $M = 179$ examples, $N = 54\,613$ features, and $D = 4$ classes; and the Bhattacharjee dataset includes $M = 203$ examples, $N = 12\,600$ features, and $D = 5$ classes. With the Sun dataset, we applied a $\log_2(\cdot)$ transformation and z-scored prior to processing, while with Bhattacharjee we simply z-scored (since the dataset included negative values).

The test-error rate was estimated as follows for each dataset. We consider a total of T "trials." For the tth trial, we 1) partition the dataset into a training subset of size $M_{\textsf{train},t}$ and a test subset of size $M_{\textsf{test},t}$, 2) design the classifier using the training subset, and 3) apply the classifier to the test subset, recording the test errors $\{e_{tm}\}_{m=1}^{M_{\textsf{test},t}}$, where $e_{tm} \in \{0,1\}$ indicates whether the mth example was in error. We then estimate the average test-error rate using the empirical average $\widehat{\mu} \triangleq M_{\textsf{test}}^{-1} \sum_{t=1}^{T} \sum_{m=1}^{M_{\textsf{test},t}} e_{tm}$, where $M_{\textsf{test}} = \sum_{t=1}^{T} M_{\textsf{test},t}$. If the test sets are constructed without overlap, we can model $\{e_{tm}\}$ as i.i.d. Bernoulli(μ), where μ denotes the true test-error rate. Then, since $\widehat{\mu}$ is Binomial($\mu, M_{\textsf{test}}$), the standard deviation (SD) of our error-rate estimate $\widehat{\mu}$ is $\sqrt{\text{var}\{\widehat{\mu}\}} = \sqrt{\mu(1-\mu)/M_{\textsf{test}}}$. Since μ is unknown, we approximate the SD by $\sqrt{\widehat{\mu}(1-\widehat{\mu})/M_{\textsf{test}}}$.

Tables V and VI show, for each algorithm, the test-error rate estimate $\widehat{\mu}$, the approximate SD $\sqrt{\widehat{\mu}(1-\widehat{\mu})/M_{\textsf{test}}}$ of the estimate, the average runtime, and two metrics for the sparsity of $\widehat{\boldsymbol{X}}$. The $\Vert\widehat{\boldsymbol{X}}\Vert_0$ metric quantifies the number of non-zero entries in $\widehat{\boldsymbol{X}}$ (i.e., absolute sparsity), while the $\widehat{K}_{99}$ metric quantifies the number of entries of $\widehat{\boldsymbol{X}}$ needed to reach 99% of the Frobenius norm of $\widehat{\boldsymbol{X}}$ (i.e., effective sparsity). We note that the reported values of $\widehat{K}_{99}$ and $\Vert\widehat{\boldsymbol{X}}\Vert_0$ represent the average over the T folds. For both the Sun and Bhattacharjee datasets, we used $T = 19$ trials and $M_{\textsf{test},t} = \lfloor M/20 \rfloor\ \forall t$.

Table V shows results for the Sun dataset. There we see that MSA-SHyGAMP gave the best test-error rate, although the other algorithms were not far behind and all error-rate estimates were within the estimator standard deviation. SPA-SHyGAMP was the fastest algorithm and MSA-SHyGAMP was the second fastest, with the remaining algorithms running 3× to 5× slower than SPA-SHyGAMP. GLMNET's weights were the sparsest according to both sparsity metrics. SPA-SHyGAMP's weights had the second lowest value of $\widehat{K}_{99}$, even though they were technically non-sparse (i.e., $\Vert\widehat{\boldsymbol{X}}\Vert_0 = 218\,452 = ND$) as expected. Meanwhile, MSA-SHyGAMP's weights were the least sparse according to the $\widehat{K}_{99}$ metric.

Table VI shows results for the Bhattacharjee dataset. In this experiment, SPA-SHyGAMP and SBMLR were tied for the best error rate, MSA-SHyGAMP was 0.5 standard-deviations worse, and GLMNET was 1.2 standard-deviations worse. However, SPA-SHyGAMP ran about twice as fast as SBMLR, and 4× as fast as GLMNET. As in the Sun dataset, SPA-SHyGAMP returned the sparsest weight matrix according to the $\widehat{K}_{99}$ metric. The sparsities of the weight matrices returned by the other three algorithms were similar to one another in both metrics. Unlike in the Sun dataset, MSA-SHyGAMP and SBMLR had similar runtimes (which is consistent with Fig. 6(b) since N is lower here than in the Sun dataset).

D. Text Classification With the RCV1 Dataset

Next we consider text classification using the Reuter's Corpus Volume 1 (RCV1) dataset [6]. Here, each sample $(y_m, \boldsymbol{a}_m)$ represents a news article, where y_m indicates the article's topic and $\boldsymbol{a}_m$ indicates the frequencies of common words in the article. The version of the dataset that we used[7] contained $N = 47\,236$

[7] http://www.csie.ntu.edu.tw/sim;cjlin/libsvmtools/datasets/multiclass.html.

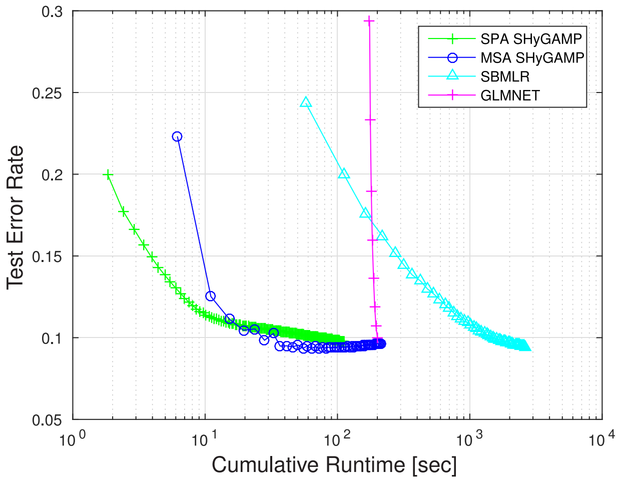


Fig. 8. Test-error rate versus runtime for the RCV1 dataset.

features and 53 topics. However, we used only the first $D = 25$ of these topics (to reduce the computational demand). Also, we retained the default training and test partitions, which resulted in the use of $M = 14\,147$ samples for training and $469\,571$ samples for testing.

The RCV1 features are very sparse (only 0.326% of the features are non-zero) and non-negative, which conflicts with the standard assumptions used for the derivation of AMP algorithms: that $\boldsymbol{A}$ is i.i.d. zero-mean and sub-Gaussian. Interestingly, the RCV1 dataset also caused difficulties for SBMLR, which diverged under default settings. This divergence was remedied by decreasing the value of a step-size parameter[8] to 0.1 from the default value of 1.

Fig. 8 shows test-error rate versus runtime for SPA-SHyGAMP, MSA-SHyGAMP, SBMLR, and GLMNET on the RCV1 dataset. In the case of SPA-SHyGAMP, MSA-SHyGAMP and SBMLR, each point in the figure represents one iteration of the corresponding algorithm. For GLMNET, each data-point represents one iteration of the algorithm *after* its CV stage has completed.[9] We used 2 CV folds (rather than the default 10) in this experiment to avoid excessively long runtimes. The figure shows that the SHyGAMP algorithms converged more than an order-of-magnitude faster than SBMLR and GLMNET, although the final error rates were similar. SPA-SHyGAMP displayed faster initial convergence, but MSA-SHyGAMP eventually caught up.

E. MNIST Handwritten Digit Recognition

Finally, we consider handwritten digit recognition using the Mixed National Institute of Standards and Technology (MNIST) dataset [42]. This dataset consists of 70 000 examples, where each example is an $N = 784$ pixel image of one of $D = 10$ digits between 0 and 9. These features were again non-negative, which conflicts with the standard AMP assumption of i.i.d. zero-mean $\boldsymbol{A}$.

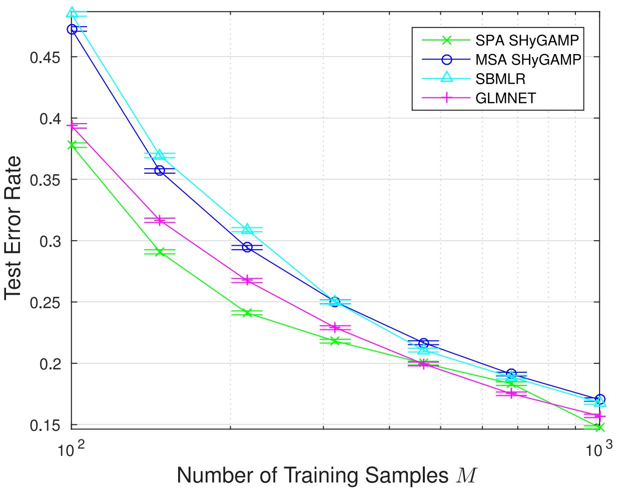


Fig. 9. Estimated test-error rate versus M for the MNIST dataset, with error-bars indicating the standard deviation of the estimate.

Our experiment characterized test-error rate versus the number of training examples, M, for the SPA-SHyGAMP, MSA-SHyGAMP, SBMLR, and GLMNET algorithms. For each value of M, we performed 50 Monte-Carlo trials. In each trial, M training samples were selected uniformly at random and the remainder of the data were used for testing. Fig. 9 shows the average estimated test-error rate $\widehat{\mu}$ versus the number of training samples, M, for the algorithms under test. The error-bars in the figure correspond to the average of the per-trial estimated SD over the 50 trials. For SBMLR, we reduced the stepsize to 0.5 from the default value of 1 to prevent a significant degradation of test-error rate. The figure shows SPA-SHyGAMP attaining significantly better error-rates than the other algorithms at small values of M (and again at the largest value of M considered for the plot). For this plot, M was chosen to focus on the $M < N$ regime.

VI. Conclusion

For the problem of multi-class linear classification and feature selection, we proposed several AMP-based approaches to sparse MLR. We started by proposing two algorithms based on HyGAMP [18], one of which finds the MAP linear classifier based on the multinomial logistic likelihood and a Laplacian prior, and the other of which finds an approximation of the test-error-rate minimizing linear classifier based on the multinomial logistic likelihood and a BG prior. The numerical implementation of these algorithms is challenged, however, by the need to solve D-dimensional inference problems of multiplicity M at each HyGAMP iteration. Thus, we proposed simplified HyGAMP (SHyGAMP) approximations based on a diagonalization of the message covariances and a careful treatment of the D-dimensional inference problems. In addition, we described EM- and SURE-based methods to tune the hyperparameters of the assumed statistical model. Finally, using both synthetic and real-world datasets, we demonstrated improved error-rate

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

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

and runtime performance relative to the state-of-the-art SBMLR [13] and GLMNET [14] approaches to sparse MLR.

References

[1] H. Sun *et al.*, "Neuronal and glioma-derived stem cell factor induces angiogenesis within the brain," *Cancer Cell*, vol. 9, pp. 287–300, 2006.

[2] A. Bhattacharjee *et al.*, "Classification of human lung carcinomas by mRNA expression profiling reveals distinctadenocarcinoma subclasses," *Proc. Nat. Acad. Sci.*, vol. 98, pp. 13790–13795, Nov. 2001.

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

[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 Cognitive Sci.*, vol. 10, pp. 424–430, Sep. 2006.

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

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

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

[8] 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.

[9] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2007.

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

[11] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink, "Sparse multinomial logistic regression: Fast algorithms and generalization bounds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, pp. 957–968, Jun. 2005.

[12] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale Bayesian logistic regression for text categorization," *Technometrics*, vol. 49, pp. 291–304, Aug. 2007.

[13] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *J. Statist. Softw.*, vol. 33, pp. 1–22, Jan. 2010.

[14] G. C. Cawley, N. L. C. Talbot, and M. Girolami, "Sparse multinomial logistic regression via Bayesian L1 regularisation," in *Proc. Neural Inf. Process. Syst. Conf.*, 2007, pp. 209–216.

[15] L. Meier, S. van de Geer, and P. Bühlmann, "The group lasso for logistic regression," *J. Roy. Statist. Soc. B*, vol. 70, pp. 53–71, 2008.

[16] Y. Grandvalet, "Least absolute shrinkage is equivalent to quadratic penalization," in *Proc. Int. Conf. Artif. Neural Netw.*, 1998, pp. 201–206.

[17] D. J. C. MacKay, "The evidence framework applied to classification networks," *Neural Comput.*, vol. 4, pp. 720–736, 1992.

[18] S. Rangan, A. K. Fletcher, V. K. Goyal, and P. Schniter, "Hybrid generalized approximate message passing with applications to structured sparsity," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2012, pp. 1236–1240.

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

[20] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Int. Symp. Inf. Theory*, pp. 2168–2172, Aug. 2011.

[21] J. Ziniel, P. Schniter, and P. Sederberg, "Binary classification and feature selection via generalized approximate message passing," *IEEE Trans. Signal Process.*, vol. 63, no. 8, pp. 2020–2032, Apr. 2015.

[22] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.*, vol. 106, pp. 18914–18919, Nov. 2009.

[23] 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.

[24] D. A. Knowles and T. P. Minka, "Non-conjugate variational message passing for multinomial and binary regression," in *Proc. Neural Inf. Process. Syst. Conf.*, 2011, pp. 1701–1709.

[25] J. P. Vila and P. Schniter, "Expectation-maximization Gaussian-mixture approximate message passing," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4658–4672, Oct. 2013.

[26] A. Mousavi, A. Maleki, and R. G. Baraniuk, "Parameterless, optimal approximate message passing," arXiv paper, arXiv:1311.0035, Nov. 2013.

[27] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artif. Intell.*, vol. 42, pp. 393–405, 1990.

[28] 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.

[29] S. Rangan, P. Schniter, E. Riegler, A. Fletcher, and V. Cevher, "Fixed points of generalized approximate message passing with arbitrary matrices," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 664–668.

[30] 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*, Jul. 2014, pp. 236–240.

[31] E. M. Byrne, "Sparse multinomial logistic regression via approximate message passing," Master's thesis, The Ohio State Univ., Columbus, OH, USA, Jul. 2015.

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

[33] D. R. Hunter and K. Lange, "A tutorial on MM algorithms," *Amer Statist.*, vol. 58, no. 1, pp. 30–37, 2004.

[34] D. Hunter and R. Li, "Variable selection using MM algorithms," *Ann. Statist.*, vol. 33, no. 4, pp. 1617–1642, 2005.

[35] D. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Scientific, 2nd ed., 1999.

[36] D. Böhning, "Multinomial logistic regression algorithm," *Ann. Inst. Statist. Math.*, vol. 44, pp. 197–200, 1992.

[37] P. Schniter, "Turbo reconstruction of structured sparse signals," in *Proc. Conf. Inf. Sci. Syst.*, Mar. 2010, pp. 1–6.

[38] L. A. Stefanski, "A normal scale mixture representation of the logistic distribution," *Statist. Probability Lett.*, vol. 11, no. 1, pp. 69–70, 1991.

[39] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[40] C. M. Stein, "Estimation of the mean of a multivariate normal distribution," *Ann. Statist.*, vol. 9, pp. 1135–1151, 1981.

[41] M. I. Jordan, "Why the logistic function? A tutorial discussion on probabilities and neural networks," Computational Cognitive Science, Massachusetts Inst. of Technol., Cambridge, MA, USA, Tech. Rep. 9503, 1995.

[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

Evan Byrne received the B.S. and M.S. degrees in electrical and computer engineering from the Ohio State University, Columbus, OH, USA, in 2012 and 2015, respectively.

His research interests include statistical signal processing, wireless communications and networks, and machine learning.

Philip Schniter (F'14) received the B.S. and M.S. degrees in electrical engineering from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 1992 and 1993, respectively, and the Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY, USA, 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 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 current research interests include statistical signal processing, wireless communications and networks, and machine learning.