Sparse Multinomial Logistic Regression via Approximate Message Passing

Evan Byrne and Philip Schniter

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 (HyGAMP) framework: one finds the maximum a posteriori (MAP) 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 (SURE) and expectation-maximization (EM), 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, belief propagation, 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 \( \{(y_m, a_m)\}_{m=1}^M \), where \( a_m \in \mathbb{R}^N \) 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 \( a_0 \). In the linear approach to this problem, the training data are used to design a weight matrix \( X \in \mathbb{R}^{N \times D} \) that generates a vector of “scores” \( z_0 \triangleq X^T a_0 \in \mathbb{R}^D \), the largest of which can be used to predict the unknown label, i.e.,

\[
\hat{y}_0 = \text{arg max}_d |z_0|_d.
\]  

In feature selection, the goal is to determine which subset of the \( N \) features \( a_0 \) 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 [12], multi-voxel pattern analysis (MVPA) [34], text mining [56], and analysis of marketing data [77].

The authors are with the Department of Electrical and Computer Engineering at The Ohio State University, Columbus, OH.

Please direct all correspondence to Prof. Philip Schniter, Dept. ECE, 2015 Neil Ave., Columbus OH 43210, e-mail: schniter@ece.osu.edu, phone 614.247.6488, fax 614.292.7506. Evan Byrne can be reached at the same address/phone/fax and e-mailed at byrne.133@osu.edu.

This work was supported in part by the National Science Foundation grants CCF-1018368 and CCF-1218754.

Portions of this work were presented at the 2015 Duke Workshop on Sensing and Analysis of High Dimensional Data.

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 \( a_m \) is i.i.d. Gaussian [8]. Note that, when \( K \ll N \), accurate linear classification can be accomplished using a sparse weight matrix \( X \), 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 \( y \triangleq [y_0, \ldots, y_M]^T \) is modeled as a realization of a random[1] vector \( y \triangleq [y_0', \ldots, y_M']^T \), the “true” weight matrix \( X \) is modeled as a realization of a random matrix \( X \), and the features \( A \triangleq [a_0, \ldots, a_M]^T \) are treated as deterministic. Moreover, the labels \( y_m \) are modeled as conditionally independent given the scores \( z_m \triangleq X^T a_m \), i.e.,

\[
\Pr\{y = y | X = X; A\} = \prod_{m=1}^M p_{y_m}(y_m | X^T a_m),
\]

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

\[
p_{y_m}(y_m | z_m) = \frac{\exp([z_m|y_m])}{\sum_{d=1}^D \exp([z_m|d])}, \quad y_m \in \{1, \ldots, D\}.
\]

The rows \( x_n \) of the weight matrix \( X \) are then modeled as i.i.d.,

\[
p_X(x) = \prod_{n=1}^N p_X(x_n),
\]

where \( p_X \) may be chosen to promote sparsity.

C. Existing methods

Several sparsity-promoting MLR algorithms have been proposed (e.g., [10][11][12][13][14][15]), differing in their choice of \( p_X \) and methodology of estimating \( X \). For example, [11][12][13] use the i.i.d. Laplacian prior

\[
p_X(x_n; \lambda) = \prod_{d=1}^D \frac{\lambda}{2} \exp(-\lambda|x_{nd}|),
\]

1For clarity, we typeset random quantities in sans-serif font and deterministic quantities in serif font.
with \( \lambda \) tuned via cross-validation. To circumvent this tuning problem, [14] employs the Laplacian scale mixture

\[
p_X(x_n) = \prod_{d=1}^D \int \left[ \frac{\lambda}{2} \exp(-\lambda|x_{nd}|) \right] p(\lambda) \, d\lambda, \quad (6)
\]

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

\[
p_X(x_n) = \prod_{d=1}^D \int N(x_{nd}; 0, \nu) p(\nu) \, d\nu, \quad (7)
\]

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 \( p(\nu; \lambda) = \frac{1}{\lambda} \exp(-\frac{1}{\lambda} \nu) \nu \geq 0 \) would lead back to the i.i.d. Laplacian distribution [5] for \( p_X \) [16]. Finally, [13] uses

\[
p_X(x_n; \lambda) \propto \exp(-\lambda \|x_n\|_2), \quad (8)
\]

which encourages row-sparsity in \( X \).

Once the probabilistic model [2-4] has been specified, a procedure is needed to infer the weights \( X \) from the training data \( \{ (y_m, a_m) \}_{m=1}^M \). The Laplacian-prior methods [11,12,13,15] use the maximum a posteriori (MAP) estimation framework:

\[
\hat{X} = \arg \max_X \log p(X|y; A) = \arg \max_X \sum_{m=1}^M \log p(y_m|X^T a_m) + \sum_{n=1}^N \log p_X(x_n), \quad (10)
\]

where Bayes’ rule was used for (10). Under \( p_X \) from (6) or (8), the second term in (10) reduces to \(-\lambda \sum_{n=1}^N \|x_n\|_1 \) or \(-\lambda \sum_{n=1}^N \|x_n\|_2 \), respectively. In this case, (10) is concave and can be maximized in polynomial time; 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 \( \lambda \) or \( \nu \) is estimated in the outer loop, and—for fixed \( \lambda \) or \( \nu \)—the resulting conjugate (i.e., \( \ell_2 \) or \( \ell_1 \) regularized) MAP optimization is solved in the inner loop.

The methods [10,11,12,13,14,15] described above all yield a sparse point estimate \( \hat{X} \). Thus, feature selection is accomplished by examining the row-support of \( \hat{X} \) and classification is accomplished through \( \hat{X} \).

**D. Contributions**

In Section III 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 \( p_{y|z} \) 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 [5] is not separable, i.e., \( p_{y|z}(y_m|z_n) \neq \prod_d p(y_m|z_{md}) \). 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 \( p_{y|z} \) 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, importance sampling, Taylor-series approximation, and a novel Gaussian-mixture approximation, and we conduct numerical experiments that suggest the superiority of the latter.

In Section 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 \( \theta \) is written as \( p_X(x; \theta) \), where the subscript and parameterization are sometimes omitted for brevity. Column vectors are typeset in boldface lower-case (e.g., \( y \) or \( y \)), matrices in boldface upper-case (e.g., \( X \) or \( X \)), and their transpose is denoted by \( (\cdot)^T \). \( E\{ \cdot \} \) denotes expectation and \( \text{Cov}\{ \cdot \} \) autocovariance. \( I_K \) denotes the \( K \times K \) identity matrix, \( e_k \) the \( k \)th column of \( I_K \), \( I_L \) the length-\( K \) vector of ones, and \( \text{Diag}(\cdot) \) the diagonal matrix created from the vector \( b \), \( |B|_{m,n} \) denotes the element in the \( m^{th} \) row and \( n^{th} \) column of \( B \), and \( \| \cdot \|_F \) the Frobenius norm. Finally, \( A \) denotes the Kronecker delta sequence, \( \delta(z) \) the Dirac delta distribution, and \( I_A \) 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
sum-product algorithm (SPA) variant of HyGAMP is a loopy belief propagation (LBP) approximation of the classification-error-rate minimizing linear classifier and that the min-sum algorithm (MSA) variant is an LBP approach to solving the MAP problem (10).

A. Classification via sum-product HyGAMP

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

\[
\hat{y}_t = \arg \max_{y_t \in \{1, \ldots, D\}} p_{y_t|y_{1:M}}(y_t | y_{1:M}; A),
\]

(11)

under known \( p_{yT} \) and \( p_a \), where \( y_{1:M} \triangleq [y_1, \ldots, y_M]^T \) and \( A \triangleq [a_1, \ldots, a_{M+T}]^T \). The probabilities in (11) can be computed via the marginalization

\[
p_{y_{1:M}}(y_{1:M}; A) = p_{y_{1:M}}(y_{1:M}; A) Z_1^{-1}
\]

(12)

\[
p_y(x, y; A) = \sum_{y \in y_{1:M}} p_y(x, y, a_{1:M+T}; A) Z_{x,y}^{-1}
\]

(13)

with scaling constant \( Z_{x,y}^{-1} \), label vector \( y = [y_1, \ldots, y_{M+T}]^T \), and constraint set \( y_t \in 1, \ldots, D \) for \( t = 1, \ldots, M \). Thus, the approximate posterior pdf used in lines 15-16 is

\[
p_{xT|y}(x_T|m; Q_{m}) = \frac{p_x(x_T|m; \hat{P}_m; Q_{m})}{\int p_x(x_T|m; \hat{P}_m; Q_{m}) \, dx_T}.
\]

(15)

A similar interpretation holds for HyGAMP’s approximation of the posterior mean and covariance of \( z_m \) in lines 15-16 which uses the intermediate vector \( \hat{p}_m(t) \) and the approximate posterior pdf

\[
p_{zT|y}(z_T|m; Q_{m}) = \frac{p_z(z_T|m; \hat{P}_m; Q_{m})}{\int p_z(z_T|m; \hat{P}_m; Q_{m}) \, dz_T}.
\]

(16)

B. Classification via min-sum HyGAMP

As discussed in Section III, an alternative approach to linear classification and feature selection is through MAP estimation of the true weight matrix \( X \). Given a likelihood of the form (3) 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 min-sum algorithm (MSA) (13) can be used to compute the MAP estimate. Although a direct application of the MSA is intractable when \( p_{yT} \) takes the MLR form (3), the MSA simplifies in the large-system limit under i.i.d. sub-Gaussian \( A \), leading to the HyGAMP approximation (18) given in Algorithm (10).

Although in practical MLR applications \( A \) is not i.i.d. Gaussian, the numerical results in Section V suggest that treating it as such works sufficiently well.

We note from Fig. 1a that the HyGAMP algorithm is applicable to a factor graph with vector-valued node variables. 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 (10). 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 \( A \), and (22) for fixed-point and local-convergence analysis of GAMP under arbitrary \( A \).

Lines (47) of Algorithm (10) produce an approximation of the posterior mean and covariance of \( x_t \) at each iteration \( t \). Similarly, lines (15) produce an approximation of the posterior mean and covariance of \( z_m \). The posterior mean and covariance of \( x_m \) are computed from the intermediate quantity \( \tilde{r}_m(t) \), which behaves like a noisy measurement of the true \( x_m \). In particular, for i.i.d. Gaussian \( A \) in the large-system limit, \( \tilde{r}_m(t) \) is a typical realization of the random vector \( r_m = x_m + v_n \) with \( v_n \sim N(0, Q_r(t)) \). Thus, the approximate posterior pdf used in lines (15) is

\[
\hat{p}_m(x_T|m; \hat{P}_m; Q_{m}) = \frac{p_x(x_T|m; \hat{P}_m; Q_{m})}{\int p_x(x_T|m; \hat{P}_m; Q_{m}) \, dx_T}.
\]

(15)

A similar interpretation holds for HyGAMP’s approximation of the posterior mean and covariance of \( z_m \) in lines (15) which uses the intermediate vector \( \hat{p}_m(t) \) and the approximate posterior pdf

\[
p_{zT|y}(z_T|m; Q_{m}) = \frac{p_z(z_T|m; \hat{P}_m; Q_{m})}{\int p_z(z_T|m; \hat{P}_m; Q_{m}) \, dz_T}.
\]

(16)

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 $\in \{\text{SPA, MSA}\}$, matrix $A$, vector $y$, pdfs $p_{x|y}$ and $p_{y|x}$ from (15)–(16), initializations $\mathbf{F}_n(0)$, $Q_n^x(0)$.

Ensure: $t \leftarrow 0$, $\mathbf{s}_n(0) \leftarrow 0$.

1. repeat
2. if MSA then for $n = 1 \ldots N$
3. $\mathbf{x}_{n}(t) \leftarrow \max_{\mathbf{x}} \log p_{x|y}(\mathbf{x}_n|\mathbf{F}_n(t-1); Q_n^x(t-1))$
4. $Q_n^x(t) \leftarrow [-\frac{1}{2} \log p_{x|y}(\mathbf{x}_n|\mathbf{F}_n(t-1); Q_n^x(t-1))]^{-1}$
5. else if SPA then for $n = 1 \ldots N$
6. $\mathbf{s}_n(t) \leftarrow E \{ \mathbf{x}_n \mid \mathbf{r}_n = \mathbf{F}_n(t-1); Q_n^x(t-1) \}$
7. $Q_n^x(t) \leftarrow \text{Cov} \{ \mathbf{x}_n \mid \mathbf{r}_n = \mathbf{F}_n(t-1); Q_n^x(t-1) \}$
8. end if
9. $\forall n: Q_n^p(t) \leftarrow \sum_{m=1}^{N} A_{nm}^2 Q_n^x(t)$
10. $\forall n: \mathbf{p}_{nm}(t) \leftarrow \sum_{k=1}^{M} A_{nm} \mathbf{s}_{nk}(t) - Q_n^p(t) \mathbf{s}_{nm}(t-1)$
11. if MSA then for $m = 1 \ldots M$
12. $\mathbf{z}_{m}(t) \leftarrow \max_{\mathbf{z}} \log p_{y|x}(\mathbf{z}|y_m, \mathbf{p}_m; Q_m^x(t))$
13. $Q_m^x(t) \leftarrow [-\frac{1}{2} \log p_{y|x}(\mathbf{z}|y_m, \mathbf{p}_m; Q_m^x(t))]^{-1}$
14. else if SPA then for $m = 1 \ldots M$
15. $\mathbf{z}_{m}(t) \leftarrow E \{ \mathbf{z}_m \mid y_m, \mathbf{p}_m = Q_m^x(t) \}$
16. $Q_m^x(t) \leftarrow \text{Cov} \{ \mathbf{z}_m \mid y_m, \mathbf{p}_m = Q_m^x(t) \}$
17. end if
18. $\forall m: Q_m^p(t) \leftarrow [Q_m^x(t)-1 - |Q_m^x(t)-1| Q_m^p(t)] |Q_m^x(t)-1|^{-1}$
19. $\forall m: \mathbf{p}_{nm}(t) \leftarrow [Q_m^x(t)-1 \mathbf{p}_m(t)] |Q_m^x(t)-1|^{-1}$
20. $\forall n: Q_n^p(t) \leftarrow \sum_{m=1}^{M} A_{nm}^2 Q_n^x(t)$
21. $\forall n: \mathbf{p}_{nm}(t) \leftarrow \mathbf{z}_m(t) + Q_n^p(t) \sum_{m=1}^{M} A_{nm} \mathbf{s}_{nk}(t)$
22. $t \leftarrow t + 1$
23. until Terminated

MAP estimation of $\mathbf{x}_n$ and line 8 as measuring the local curvature of the corresponding MAP cost. Similar interpretations hold for MAP estimation of $\mathbf{z}_m$ via lines 12–13.

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 3–4 and 12–16. We now briefly discuss the latter problems for the sum-product version of HyGAMP.

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

$$p_{x|y}(\mathbf{x}_n) = (1 - \beta) \delta(\mathbf{x}_n) + \beta N(\mathbf{x}_n; 0, vI),$$

(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 6–8 of Algorithm 1 reduce to

$$C_m = 1 + \frac{1 - \beta}{\beta} N(0; \mathbf{r}_m, Q_m^x)$$

(18)

$$\mathbf{z}_m = C_m^{-1} (I + v^{-1} Q_m^x)^{-1} \mathbf{r}_m$$

(19)

$$Q_m^x = C_m^{-1} (I + v^{-1} Q_m^x)^{-1} Q_m^x + (C_n - 1) \mathbf{z}_m^T \mathbf{z}_m,$$

(20)

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

2) Inference of $\mathbf{z}_m$: When $p_{y|x}$ takes the MLR form in (3), closed-form expressions for $\mathbf{z}_m(t)$ and $Q_m^x(t)$ from lines 15–16 of Algorithm 1 do not exist. While these computations could be approximated using, e.g., numerical integration or importance sampling, this is expensive because $\mathbf{z}_m(t)$ and $Q_m^x(t)$ 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 $\mathbf{x}_n$: To ease the computation of line 3 in Algorithm 1 it is typical to choose a log-concave prior $p_{x|y}$ so that the optimization problem (10) is concave (since $p_{y|x}$ in (3) is also log-concave). As discussed in Section III-C, a common example of a log-concave sparsity-promoting prior is the Laplace prior (5). In this case, line 3 becomes

$$\hat{\mathbf{x}}_n = \arg \max_x -\frac{1}{2} (\mathbf{x} - \mathbf{r}_n)^T Q_n^x^{-1} (\mathbf{x} - \mathbf{r}_n) - \lambda \| \mathbf{x} \|_1,$$

(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 $J(x)$, MM iterates the recursion

$$\hat{x}^{(k+1)} = \arg \max_x J(x; \hat{x}^{(k)}),$$

(22)

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

$$\hat{x}^{(k+1)} = \arg \max_x \hat{J}_n(x; \hat{x}(k))$$

(23)

$$= \arg \max_x \mathbf{z}^T (Q_n^x)^{-1} \mathbf{r}_n - \frac{1}{2} \mathbf{z}^T (Q_n^x)^{-1} + \lambda \| x - \hat{x}(k) \|_2^2) x$$

(24)

$$= ((Q_n^x)^{-1} + \lambda \| \mathbf{x} \|_2^2)^{-1} Q_n^x^{-1} \mathbf{r}_n$$

(25)

where (24) dropped the $x$-invariant terms from $\hat{J}_n(x; \hat{x}(k))$. 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 $Q_n^x$ equal to the Hessian of the objective function in (21) at $\hat{x}_n$. Recalling that the second derivative of $[x_{nd}]$ is undefined when $x_{nd} = 0$ but otherwise equals zero, we set $Q_n^x = Q_n^y$ but then zero the $dh$ row and column of $Q_n^x$ for all $d$ such that $\hat{x}_{nd} = 0$.

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

$$\hat{\mathbf{z}}_m = \arg \max_{\mathbf{z}} -\frac{1}{2} (\mathbf{z} - \mathbf{p}_m)^T Q_m^p(\mathbf{z} - \mathbf{p}_m)$$

+ log $p_{y|x}(\mathbf{y}_n | \mathbf{z})$.

(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.,

$$\hat{z}^{(k+1)} = \hat{z}^{(k)} - \alpha^{(k)} H^{(k)^{-1}} g^{(k)},$$

(27)

where $g^{(k)}$ and $H^{(k)}$ are the gradient and Hessian of the objective function in (26) at $\hat{z}^{(k)}$, and $\alpha^{(k)} \in (0, 1]$ is a
stepsize. From (35), it can be seen that
\[ \frac{\partial}{\partial z} \log p(y|z) = \delta_{y,z} - \partial_y p(y,z), \]
and so
\[ g_m^{(k)} = u(z_m^{(k)}) - e_m + [Q_m^{(k)}]^{-1}(z_m^{(k)} - \bar{p}_m), \]
where \( e_m \) denotes the \( m \)th column of \( I_D \) and \( u(z) \in \mathbb{R}^{D \times 1} \) is defined elementwise as
\[ u(z)_k \equiv p(y_k|z). \]
Similarly, it is known [36] that the Hessian takes the form
\[ H_m^{(k)} = u(z_m^{(k)})u(z_m^{(k)})^T - \text{Diag}(u(z_m^{(k)})) - [Q_m^{(k)}]^{-1}, \]
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.,
\[ z_m^{(k+1)} = z_m^{(k)} - \alpha_m g_m^{(k)} H_m^{(k)} x, \]
where \( g_m^{(k)} \) and \( H_m^{(k)} \) are the first and second derivatives of the objective function in (29) with respect to \( z_d \) at \( z = z_m^{(k)} \). From (28)-(30), it follows that
\[ g_m^{(k)} = \partial_y p(y_m|z) - \delta_{y_m,y} + \left([Q_m^{(k)}]^{-1}\right)_{d,d}(z_m^{(k)} - \bar{p}_m), \]
\[ H_m^{(k)} = \partial_y p(y_m|z)^2 - \partial_y p(y_m|z) - \left([Q_m^{(k)}]^{-1}\right)_{d,d}. \]

E. HyGAMP summary

In summary, the SPA and MSA variants of the HyGAMP algorithm provide tractable methods of approximating the posterior test-label probabilities \( p(y_1, y_2|y_1: M; A) \) and computing the MAP weight matrix \( \hat{X} = \arg\max_X p(y_1, y_2|y_1: M; A) \), respectively, under a separable likelihood and a separable prior. In particular, HyGAMP attacks the high-dimensional inference problems of interest using a sequence of low-dimensional (in particular, \( D \)-dimensional) inference problems and \( D \times D \) matrix inversions, as detailed in Algorithm 1.

As detailed in the previous subsections, 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 III, compares favorably with existing state-of-the-art methods.

### III. SHyGAMP for Multiclass Classification

As described in Section III, 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 \( Q_n^s, Q_n^p, Q_m^p, \) and \( Q_m^z \) to be diagonal. In other words,
\[ Q_n^s = \text{Diag} \{ q_{n1}, \ldots, q_{nD} \}, \]
and similar for \( Q_n^p, Q_m^p, \) and \( Q_m^z \). As a consequence, the \( D \times D \) matrix inversions in lines 18 and 20 of Algorithm 1 reduce to \( D \) scalar inversions. More importantly, the \( D \)-dimensional inference problems in lines 18 and 20 can be tackled using much simpler methods than those described in Section III 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 \( \{q_{nd}\} \) by a value invariant to both \( n \) and \( d \), i.e.,
\[ q^x \approx \frac{1}{ND} \sum_{n=1}^N \sum_{d=1}^D q_{nd}^x. \]
Then, in line 9 in Algorithm 1 we use the approximation
\[ q_{md}^p \approx \sum_{n=1}^N A_{mn}^2 q^x \approx \frac{\|A\|_F^2}{M} q^x \approx q^p. \]

The approximation (a), after precomputing \( \|A\|_F^2 \), reduces the complexity of line 9 from \( O(ND) \) to \( O(1) \). We next define
\[ q^s \approx \frac{1}{MD} \sum_{m=1}^M \sum_{d=1}^D q_{md}^s, \]
and in line 20 we use the approximation
\[ q_{nd}^f \approx \left( \sum_{m=1}^M A_{mn}^2 q^s \right)^{-1} \approx \frac{N}{q^s \|A\|_F^2} q^f. \]

The complexity of line 20 then simplifies to \( O(MD) \) to \( O(1) \). For clarity, we note that after applying the scalar variance approximation, we have \( Q_n^s = q^s I_D \forall n \), and similar for \( Q_n^p, Q_m^p, \) and \( Q_m^z. \)

| Algorithm       | Quantity | Method | Complexity     |
|-----------------|----------|--------|----------------|
| SPA-HyGAMP      | \( z \)  | CF     | \( O(D^3) \)   |
| MSA-HyGAMP      | \( z \)  | MM     | \( O(KD^3) \)  |

### 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. 'CF' = 'closed form', 'NI' = 'numerical integration', 'MM' = 'minorization-maximization', and 'CWN' = 'component-wise Newton’s method'. For the MSA method, \( K \) denotes the number of samples per dimension, and for the MM and CWN methods \( K \) denotes the number of iterations.
B. Sum-product SHyGAMP: Inference of \( x_m \)

With diagonal \( Q_n^p \) and \( Q_n^q \), the implementation of lines 48, 62 is greatly simplified by choosing a sparsifying prior \( p_k \) with the separable form \( p_k(x_{nd}) = \prod_{d=1}^D p_k(x_{nd}) \). A common example is the Bernoulli-Gaussian (BG) prior

\[
p_k(x_{nd}) = (1 - \beta_d)\delta(x_{nd}) + \beta_dN(x_{nd}; m_d, v_d I). \tag{39}\]

For any separable \( p_k \), lines 51, 63 reduce to computing the mean and variance of the distribution

\[
p_{k|x}(x_{nd}|\tilde{r}_{nd}, q_{nd}) = \frac{p_k(x_{nd})N(x_{nd}; \tilde{r}_{nd}, q_{nd})}{\int p_k(x_{nd})N(x_{nd}; \tilde{r}_{nd}, q_{nd}) \, dx_{nd}}. \tag{40}\]

for all \( n = 1 \ldots N \) and \( d = 1 \ldots D \), as in the simpler GAMP algorithm 45. With the BG prior 39, these quantities can be computed in closed form (see, e.g., 37).

C. Sum-product SHyGAMP: Inference of \( z_m \)

With diagonal \( Q_m^p \) and \( Q_m^q \), the implementation of lines 13, 16 can also be greatly simplified. Essentially, the problem becomes that of computing the scalar means and variances

\[
\hat{z}_{md} = C_m^{-1}\int_{\mathbb{R}^D} z_d p_{y|z}(y_m|z) \prod_{k=1}^D N(z_k; \hat{p}_{mk}, q_{mk}^p) \, dz \tag{41}
\]

\[
\hat{q}_{md}^2 = C_m^{-1}\int_{\mathbb{R}^D} z_d^2 p_{y|z}(y_m|z) \prod_{k=1}^D N(z_k; \hat{p}_{mk}, q_{mk}^p) \, dz - \hat{z}_{md}^2 \tag{42}
\]

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

\[
C_m \triangleq \int_{\mathbb{R}^D} p_{y|z}(y_m|z) \prod_{k=1}^D N(z_k; \hat{p}_{mk}, q_{mk}^p) \, dz. \tag{43}
\]

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

1) Numerical integration: A straightforward approach to (approximately) computing 41-42 is through numerical integration (NI). For this, we propose to use a hyper-rectangular grid of \( z \) values where, for \( z_d \), the interval \( \tilde{z}_{md} = \alpha D q_{md}^p, \tilde{z}_{md} + \alpha D q_{md}^p \) is sampled at \( K \) equispaced 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 41-42 can be obtained through importance sampling (IS) 9 §11.1.4. Here, we draw \( K \) independent samples \( \{\tilde{z}_m[k]\}_{k=1}^K \) from \( N(\hat{p}_m, Q_m^p) \) and compute

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

\[
\hat{z}_{md} \approx C_m^{-1}\sum_{k=1}^K \tilde{z}_{md}[k] p_{y|z}(y_m|\tilde{z}_m[k]) \tag{45}
\]

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

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 \( p_{y|z} \) using a second-order Taylor series (TS) about \( \hat{p}_m \), i.e., \( p_{y|z}(y_m|z) \approx f_m(z; \hat{p}_m) \) with

\[
f_m(z; \hat{p}_m) \triangleq \frac{\partial}{\partial z} p_{y|z}(y_m|z) \bigg|_{z=\hat{p}_m} \tag{47}
\]

and Hessian \( H_m(\hat{\mathbf{p}}) \triangleq \left( \frac{\partial^2}{\partial z^2} p_{y|z}(y_m|z) \right) \bigg|_{z=\hat{p}_m} \). In this case, it can be shown 31 that

\[
C_m \approx f_m(\hat{\mathbf{p}}) + \frac{1}{2} \sum_{k=1}^D H_{mk}(\hat{\mathbf{p}}) q_{mk}^p \tag{48}
\]

\[
\hat{z}_{md} \approx \hat{C}_m^{-1}\left( f_m(\hat{\mathbf{p}}) \hat{p}_{md} + g_{md}(\hat{\mathbf{p}}) q_{md}^p \right) \tag{49}
\]

\[
q_{md}^2 \approx C_m^{-1}\left( f_m(\hat{\mathbf{p}}) (\hat{p}_{md}^2 + g_{md}(\hat{\mathbf{p}})) + 2g_{md}(\hat{\mathbf{p}}) \hat{p}_{md} q_{md}^p \right. \right. \]

\[
\left. + \frac{1}{2} \sum_{k=1}^D \hat{p}_{mk} q_{mk}^p H_{mk}(\hat{\mathbf{p}}) \right) \tag{50}
\]

where \( H_{md}(\hat{\mathbf{p}}) \triangleq \left[ H_m(\hat{\mathbf{p}}) \right]_{dd} \). 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 41-42 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 Gaussian mixture (GM) approximation, we work with the difference variables

\[
\gamma_d(y) \triangleq \begin{cases} \frac{z_y - z_d}{z_y} & d \neq y \\ z_y & d = y \end{cases} \tag{51}
\]
Their utility can be seen from the fact that (recalling (53))

\[ p_{y|z}(y|z) = \frac{1}{1 + \sum_{d \neq y} \exp(z_d - z_y)} \]

\[ = \frac{1}{1 + \sum_{d \neq y} \exp(-\gamma_d)} \approx t^y(\gamma(y)), \]

which is smooth, positive, and bounded by 1, and strictly increasing in \( \gamma_d \). Thus, \( \gamma_d \) for appropriately chosen \( \{\alpha_t, \mu_kl, \sigma_kl\} \).

\[ t^y(\gamma) \approx \sum_{l=1}^{L} \alpha_l \prod_{k \neq y} \Phi \left( \frac{\gamma_k - \mu_k}{\sigma_k} \right) \approx \hat{t}^y(\gamma), \]

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 \( \sup_{\gamma \in \mathbb{R}} |t^y(\gamma) - \hat{t}^y(\gamma)| \).

Recall from (41)-(43) that our objective is to compute \( S_{di}^{(y)} \)

\[ S_{di}^{(y)} = \int_{\mathbb{R}^D} (e_d^T z)^i p_{y|z}(y|z) N(z; \hat{\mu}, \hat{Q}^P) \, dz \approx S_{di}^{(y)}, \]

where \( i \in \{0, 1, 2\} \), \( Q^P \) is diagonal, and \( e_d \) is the \( d \)th column of \( I_D \). To exploit (54), we change the integration variable to

\[ \gamma(y) = T_y z \]

with

\[ T_y = \begin{bmatrix} -I_{y-1} & 1_{(y-1) \times 1} & 0_{(y-1) \times (D-y)} \\ 0_{1 \times (y-1)} & 1 & 0_{1 \times (D-y)} \\ 0_{(D-y) \times (y-1)} & 1_{(D-y) \times 1} & -I_{D-y} \end{bmatrix} \]

(57)

to get (since \( \det(T_y) = 1 \))

\[ S_{di}^{(y)} = \int_{\mathbb{R}^D} (e_d^T T_y^{-1} \gamma)^i t^y(\gamma) N(\gamma; T_y \hat{\mu}, T_y \hat{Q}^P T_y^T) \, d\gamma. \]

(58)

Then, applying the approximation (54) and

\[ N(\gamma; T_y \hat{\mu}, T_y \hat{Q}^P T_y^T) = N(\gamma(y); \hat{\gamma}_y, \hat{Q}_y^P) \times \prod_{k \neq y} N(\gamma_k; \gamma_y - \hat{\mu}_k, \hat{Q}_k) \]

(59)

to (58), we find that

\[ S_{di}^{(y)} \approx \sum_{l=1}^{L} \alpha_l \int_{\mathbb{R}} N(\gamma; \hat{\mu}, \hat{Q}_y^P) \left[ \int_{\mathbb{R}^{D-1}} (e_d^T T_y^{-1} \gamma)^i \times \prod_{k \neq y} \Phi \left( \frac{\gamma_k - \mu_k}{\sigma_k} \right) \right] d\gamma, \]

(60)

Noting that \( T_y^{-1} = T_y \), we have

\[ e_d^T T_y^{-1} \gamma = \begin{cases} \gamma_y - \gamma_d & d \neq y \\ \gamma_y & d = y \end{cases}. \]

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

\[ \int_{\mathbb{R}} t^i N(\gamma; c - \hat{\mu}, \hat{q}) \Phi \left( \frac{\gamma - \mu}{\sigma} \right) \, d\gamma \approx T_i \]

with \( i \in \{0, 1, 2\} \), which can be computed in closed form. In particular, defining \( x = \frac{c - \hat{\mu}}{\sqrt{\sigma^2 + \hat{q}}}, \)

\[ T_0 = \Phi(x), \]

\[ T_1 = (c - \hat{\mu}) \Phi(x) + \frac{\hat{q} \phi(x)}{\sqrt{\sigma^2 + \hat{q}}}, \]

\[ T_2 = T_1^2 \frac{\hat{q}}{\phi(x)} + q \Phi(x) - \frac{q^2 \phi(x)}{\phi(x) \Phi(x)} \]

(65)

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

If a grid of \( K \) values is used for numerical integration over \( \gamma_y \) in (60), then the overall complexity of the method grows as \( O(MDKL) \). 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 15-16 in Algorithm 1 under diagonal \( Q^P \) and \( Q^T \). 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 \( \hat{p} \) and \( Q^P \)):

1) generate i.i.d. samples \( z_{true}[t] \sim N(z; \hat{\mu}, \hat{Q}^P) \) and \( y_{true}[t] \sim p_{y|z}(y|z_{true}[t]) \) for \( t = 1 \ldots T \),

2) compute the approximation \( \hat{z}[t] \approx E[z \mid y = y_{true}[t], \hat{p}, \hat{Q}^P] \) using each method described in Sections III-C1~III-C4

3) compute average MSE \( \hat{\gamma}_y \approx \frac{1}{T} \sum_{t=1}^{T} \|z_{true}[t] - \hat{z}[t]\|^2 \)

for each method, and we measured the combined runtime of lines [15] [16] for each method. Unless otherwise noted, we used \( D = 4 \) classes, \( \hat{p} = e_1, \hat{Q}^P = q^P I_P, \) and \( q^P = 1 \) in our experiments. For numerical integration (NI), we used a grid of size \( K = 7 \) and radius of \( \alpha = 4 \) standard deviations; for importance sampling (IS), we used \( K = 1500 \) samples; and for the Gaussian-mixture (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.

Figure 3 plots the normalized MSE versus variance \( q^P \) for the four methods under test, in addition to the trivial method \( \hat{z}[t] = \hat{p} \). The figure shows that the NI, IS, and GM methods performed similarly across the full range of \( q^P \) and always outperform the trivial method. The Taylor-series method, however, breaks down when \( q^P > 1 \). A close examination of the figure reveals that GM gave the best accuracy, IS the second best accuracy, and NI the third best accuracy.

Figure 3 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 Taylor-series method was the fastest, we saw in Fig. 2 that it is accurate only at small variances \( q^P \). Figure 3 then shows GM was about an
D. Min-sum ShyGAMP: Inference of \( x_n \)

With diagonal \( Q^p_m \) and \( Q^z_m \), the implementation of lines 13 in Algorithm 1 can be significantly simplified. Recall that, when the prior \( p_x \) is chosen as i.i.d. Laplace (5), line 5 manifests as (21), which is in general a non-trivial optimization problem. But with diagonal \( Q^p_m \), (21) decouples into \( D \) instances of the scalar optimization

\[
x_{nd} = \arg \max_x -\frac{1}{2} \left( x - \hat{r}_{nd} \right)^2 - \lambda |x|.
\]

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

\[
\hat{x}_{nd} = \sgn(\hat{r}_{nd}) \max \{0, |\hat{r}_{nd}| - \lambda q^r_{nd}\}.
\]

Above, \( \sgn(r) = 1 \) when \( r \geq 0 \) and \( \sgn(r) = -1 \) when \( r < 0 \). Meanwhile, line 4 reduces to

\[
q^r_{nd} = \left[ \frac{\partial^2}{\partial x^2} \left( \frac{1}{2} \left( x - \hat{r}_{nd} \right)^2 + \lambda |x| \right) \right]_{x = \hat{x}_{nd}}^{-1},
\]

which equals \( q^r_{nd} \) when \( \hat{x}_{nd} \neq 0 \) and is otherwise undefined. When \( \hat{x}_{nd} = 0 \), we set \( q^r_{nd} = 0 \).

E. Min-sum ShyGAMP: Inference of \( z_m \)

With diagonal \( Q^p_m \) and \( Q^z_m \), the implementation of lines 13 in Algorithm 1 also simplifies. Recall that, when the likelihood \( p_{y|z} \) 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 \( Q^p_m \) and \( Q^z_m \). When \( Q^p_m \) is diagonal, the first and second derivatives (32)-(33) reduce to

\[
q^p_{md} = p_{y|z}(d|\hat{z}^p_m) - \delta_{ym-d} + \left( \hat{z}^{(k)}_{md} - \bar{\mu}_{md} / q_{md} \right) / q^p_{md},
\]

\[
H^p_{md} = p_{y|z}(d|\hat{z}^{(k)}_m)^2 - p_{y|z}(d|\hat{z}^{(k)}_m)^2 - 1 / q^p_{md},
\]

which leads to a reduction in complexity.

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

\[
q^z_{md} = \frac{1}{1 / q^p_{md} + p_{y|z}(d|\hat{z}_m) - p_{y|z}(d|\hat{z}_m)^2}.
\]

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 cross-validation (CV) is the customary approach to tuning such model parameters, it can be very computationally costly, since each parameter must be tested over a grid of hypothesized values and over multiple data folds. For example, K-fold cross-validation tuning of P parameters using G 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 Bernoulli-Gaussian prior in (39), which has parameters \( \beta_d, m_d, \) 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 \( \beta_d, m_d, \) 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 \( \lambda. \) For this, we use a modification of the SURE-AMP framework from [26], which adjusts \( \lambda \) to minimize the Stein’s unbiased risk estimate (SURE) of the weight-vector MSE.

We describe our method by first reviewing SURE and SURE-AMP. First, suppose that the goal is to estimate the (SURE) of the weight-vector MSE. The implication of (73) is that, given only the noisy observation \( r, \) which is a realization of a random variable \( x, \) from the noisy observation \( r, \) which is a realization of

\[
r = x + \sqrt{q^r} w,
\]

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

\[
E \{ (\tilde{x} - x)^2 \} = q^r + E \left\{ g^2(r, q^r; \theta) + 2q^r g'(r, q^r; \theta) \right\}.
\]

(73)

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

\[
\text{SURE}(r, q^r; \theta) \equiv q^r + g^2(r, q^r; \theta) + 2q^r g'(r, q^r; \theta)
\]

of the MSE(\( \theta)) \equiv E \{ (\tilde{x} - x)^2 \} that is unbiased, i.e.,

\[
E \{ \text{SURE}(r, q^r; \theta) \} = \text{MSE}(\theta).
\]

These unbiased risk estimates can then be used as a surrogate for the true MSE when tuning \( \theta. \)

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

\[
\hat{\lambda} = \arg \min_{\lambda} \sum_{n=1}^N g^2(\hat{r}_n, q^r; \lambda) + 2q^r g'(\hat{r}_n, q^r; \lambda).
\]

(76)

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

\[
g^2(\hat{r}_n, q^r; \lambda) = \left\{ \begin{array}{ll}
\lambda^2(q^r)^2 & \text{if } |\hat{r}_n| > \lambda q^r \\
\frac{\lambda^2}{n} & \text{otherwise}
\end{array} \right.
\]

(77)

\[
g'(\hat{r}_n, q^r; \lambda) = \left\{ \begin{array}{ll}
-1 & \text{if } |\hat{r}_n| < \lambda q^r \\
0 & \text{otherwise}
\end{array} \right.
\]

(78)

However, solving (75) for \( \lambda \) 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 \( \lambda \) for min-sum SHyGAMP. But, instead of the empirical average in (76), we propose to use a statistical average, i.e.,

\[
\hat{\lambda} = \arg \min_{\lambda} E \{ g^2(r, q^r; \lambda) + 2q^r g'(r, q^r; \lambda) \} = \frac{1}{n}\int J(\lambda)
\]

by modeling the random variable \( r \) as a Gaussian mixture (GM) whose parameters are fitted to \( \{r_{nd}\}. \) 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 \( \lambda \) from (79) is the unique root of \( \frac{d}{d\lambda} J(\lambda). \) To find this root, we use the bisection method. In particular, due to (77)–(78), the objective in (79) becomes

\[
J(\lambda) = \int_{-\infty}^{-\lambda q^r} p_r(r)\lambda^2(q^r)^2 dr + \int_{-\lambda q^r}^{\lambda q^r} p_r(r)(r^2 - 2q^r) dr + \int_{\lambda q^r}^{\infty} p_r(r)\lambda^2(q^r)^2 dr,
\]

(80)

from which it can be shown that

\[
\frac{d}{d\lambda} J(\lambda) = 2\lambda(q^r)^2\left[1 - Pr\{ -\lambda q^r < r < \lambda q^r\} \right] - \left[p_r(\lambda q^r) + p_r(-\lambda q^r)\right]2(q^r)^2.
\]

(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 \( \lambda \) 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][11][12]. As described in Section [11][12], both SBMLR and GLMNET...
use $\ell_1$ regularization, but SBMLR tunes the regularization parameter $\lambda$ using evidence maximization while GLMNET tunes it using cross-validation (using the default value of 10 folds unless otherwise noted). For SBMLR and GLMNET, we ran code written by the authors\footnote{SBMLR obtained from \url{http://theoval.cmp.uea.ac.uk.matlab/}} 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 $\hat{X}$ 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 $a_m$ from the class-conditional generative distribution $a_m \mid y_m \sim N(\mu_{y_m}, vI_N)$. 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, $\mu_x \in \mathbb{R}^N$. In particular, we generated $[\mu_1, \ldots, \mu_D]$ by drawing a $K \times K$ matrix with i.i.d. $N(0, 1)$ entries, performing a singular value decomposition, and zero-padding the first $D$ left singular vectors to length $N$. We note that our generation of $y, A, X$ is matched [41] to the multinomial logistic model [2]-[5].

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

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

$$\Pr\{\text{cor}\mid y\} = \Pr \left[ \bigcap_{d \neq y} \left\{ \left( \bar{x}_y - \bar{x}_d \right)^T a < \left( \bar{x}_y - \bar{x}_d \right)^T \mu_y \right\} \right],$$

where $a \sim N(0, v I_N)$ and the multivariate normal cdf in (83) was computed using Matlab’s $\text{mvncdf}$.\footnote{GLMNET obtained from \url{http://www.stanford.edu/~hastie/glmnet_matlab/}}

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 $N$; 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.

Table IV: Configurations of the synthetic-data experiments.

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

Figures 4a-b show the expected test-error rate and runtime, respectively, versus feature-vector sparsity, $K$, averaged over 12 independent trials. Figure 5a shows that, at all tested values of $K$, SPA-SHyGAMP gave the best error-rates and MSA-SHyGAMP gave the second best error-rates, although those reached by GLMNET were similar at large $M$. Moreover, the error-rates of SPA-SHyGAMP, MSA-SHyGAMP, and GLMNET all converged towards the BER as $M$ increased, whereas that of SBMLR did not. Since MSA-SHyGAMP, GLMNET, and SBMLR all solve the same $\ell_1$-regularized MLR problem, the difference in their error-rates can be attributed to the difference in their tuning of the regularization parameter $\lambda$. Figure 5b 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 $M$.

Figures 6a-b show the expected test-error rate and runtime, respectively, versus the number of features, $N$, averaged over 12 independent trials. Figure 6a shows that, at all tested values of $N$, MSA-SHyGAMP gave lower error-rates than SBMLR and GLMNET. Meanwhile, SPA-SHyGAMP gave the lowest error-rates for certain values of $N$. Figure 6b shows that SPA-SHyGAMP and MSA-SHyGAMP gave the fastest runtimes. All runtimes were approximately invariant to $K$.

Figures 7a-b show the expected test-error rate and runtime, respectively, versus the number of features, $N$, averaged over 12 independent trials. Figure 7a shows that, at all tested values of $N$, MSA-SHyGAMP gave lower error-rates than SBMLR and GLMNET. Meanwhile, SPA-SHyGAMP gave the lowest error-rates for certain values of $N$. Figure 7b shows that SPA-SHyGAMP and MSA-SHyGAMP gave the fastest runtimes for $N \geq 10000$, while SBMLR gave the fastest runtimes for $N \leq 3000$. All runtimes increased linearly with $N$.

B. Example of SURE tuning

Although the good error-rate performance of MSA-SHyGAMP in Section V-A suggests that the SURE $\lambda$-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, $N = 30000$ features, $M = 300$ examples, and sparsity $K = 25$, we ran MSA-SHyGAMP using various fixed values of $\lambda$. In the sequel, we refer to this experiment as “Synthetic Experiment 4.” The resulting expected test-error rate versus $\lambda$ (averaged over 10 independent realizations) is shown in Fig. 7a. For the same realizations, we ran MSE-SHyGAMP with SURE-tuning and plot the resulting error-rate and average $\lambda$ in Fig. 7b. From Fig. 7a we see that the SURE $\lambda$-tuning method matched both the minimizer and the minimum of the error-versus-$\lambda$ trace of fixed-$\lambda$ MSA-SHyGAMP.
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 i) to determine which subset of genes best predicts the various diseases and ii) to classify whether an (undiagnosed) patient is at risk for any of these diseases based on their gene profile.

We tried two datasets: one from Sun et al. [1] and one from Bhattacharjee et al. [2]. The Sun dataset includes $M = 179$ examples, $N = 54613$ features, and $D = 4$ classes; and the Bhattacharjee dataset includes $M = 203$ examples, $N = 12600$ 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 $t$th trial, we i) partition the dataset into a training subset of size $M_{\text{train},t}$ and a test subset of size $M_{\text{test},t}$, ii) design the classifier using the training subset, and iii) apply the classifier to the test subset, recording the test errors $\{e_{tm}\}_{m=1}^M$, where $e_{tm} \in \{0, 1\}$ indicates whether the $m$th example was in error. We then estimate the average test-error rate using the empirical average $\hat{\mu} = \frac{1}{M_{\text{test}}} \sum_{t=1}^T \sum_{m=1}^{M_{\text{test}}} e_{tm}$, where $M_{\text{test}} = \sum_{t=1}^T M_{\text{test},t}$. If the test sets are constructed without overlap, we can model $\{e_{tm}\}$ as i.i.d. Bernoulli($\mu$), where $\mu$ denotes the true test-error rate. Then, since $\hat{\mu}$ is Binomial($\mu, M_{\text{test}}$), the standard deviation (SD) of our error-rate estimate $\hat{\mu}$ is $\sqrt{\text{var}(\hat{\mu})} = \sqrt{\frac{\mu(1-\mu)}{M_{\text{test}}}}$. Since $\mu$ is unknown, we approximate the SD by $\sqrt{\hat{\mu}(1-\hat{\mu})/M_{\text{test}}}$.

Tables V and VI show, for each algorithm, the test-error rate estimate $\hat{\mu}$, the approximate SD $\sqrt{\hat{\mu}(1-\hat{\mu})/M_{\text{test}}}$ of the estimate, the average runtime, and two metrics for the sparsity of $\hat{X}$. The $\|\hat{X}\|_0$ metric quantifies the number of non-zero entries in $\hat{X}$ (i.e., absolute sparsity), while the $K_{99}$ metric quantifies the number of entries of $\hat{X}$ needed to reach 99% of the Frobenius norm of $\hat{X}$ (i.e., effective sparsity). We note that...
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.

Fig. 7: Synthetic experiment 4: expected test-error rate versus regularization parameter $\lambda$ for fixed-$\lambda$ MSA-SHyGAMP. Here, $D = 4$, $M = 300$, $N = 30000$, and $K = 25$. Also shown is the average test-error rate for SURE-tuned MSA-SHyGAMP plotted at the average value of $\hat{\lambda}$.

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

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Algorithm & $\%$ Error (SD) & Runtime (s) & $K_{99}$ & $\|\hat{X}\|_0$ \\
\hline
SPA-SHyGAMP & 33.3 (3.8) & 6.86 & 20.05 & 218452 \\
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) & 31.05 & 10.89 & 16.84 \\
\hline
\end{tabular}
\caption{Estimated test-error rate, standard deviation of estimate, runtime, and sparsities for the Sun dataset.}
\end{table}

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 $K_{99}$ 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. 6b 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. Here, each sample $(y_m, a_m)$ represents a news article, where $y_m$ indicates the article’s topic and $a_m$ indicates the frequencies of common words in the article. The version of the dataset that we used contained $N = 47236$ 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 $M = 14147$ samples for training and 469571 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 $A$ 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 to 0.1 from the default value of 1.

Figure shows test-error rate versus runtime for SPA-SHyGAMP, MSA-SHyGAMP, SBMLR, and GLMNET on the RCV1 dataset. In the case of SPA-SHyGAMP, MSA-SHyGAMP and SBMLR, each point in the figure represents one iteration of the corresponding algorithm. For GLMNET, each data-point represents one iteration of the algorithm after its cross-validation stage has completed. 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. 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 \( \mathbf{A} \).

Our experiment characterized test-error rate versus the number of training examples, \( M \), for the SPA-SHyGAMP, MSA-SHyGAMP, SBMLR, and GLMNET algorithms. For each

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

| Algorithm          | % Error (SD) | Runtime (s) | \( K_{90} \) | \( \| \mathbf{X} \|_0 \) |
|--------------------|--------------|-------------|-------------|----------------|
| SPA-SHyGAMP        | 9.5 (2.1)    | 3.26        | 16.15       | 63,000        |
| MSA-SHyGAMP        | 10.5 (2.2)   | 6.11        | 55.20       | 84.65         |
| SBMLR              | 9.5 (2.1)    | 6.65        | 44.25       | 79.10         |
| GLMNET             | 12.0 (2.4)   | 13.67       | 49.65       | 89.40         |

For the problem of multi-class linear classification and feature selection, we proposed several AMP-based approaches to sparse multinomial logistic regression. We started by proposing two algorithms based on HyGAMP, one of which finds the maximum a posteriori (MAP) linear classifier based on the multinomial logistic likelihood and a Laplacian prior, and the other of which finds an approximation of the test-error rate minimizing linear classifier based on the multinomial logistic likelihood and a Bernoulli-Gaussian prior. The numerical implementation of these algorithms is challenged, however, by the need to solve \( D \)-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 and runtime performance relative to the state-of-the-art SBMLR and GLMNET approaches to sparse MLR.

VI. CONCLUSION

For the problem of multi-class linear classification and feature selection, we proposed several AMP-based approaches to sparse multinomial logistic regression. We started by proposing two algorithms based on HyGAMP, one of which finds the maximum a posteriori (MAP) linear classifier based on the multinomial logistic likelihood and a Laplacian prior, and the other of which finds an approximation of the test-error-rate minimizing linear classifier based on the multinomial logistic likelihood and a Bernoulli-Gaussian prior. The numerical implementation of these algorithms is challenged, however, by the need to solve \( D \)-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 and runtime performance relative to the state-of-the-art SBMLR and GLMNET approaches to sparse MLR.
