Sparse Multinomial Logistic Regression via Approximate Message Passing

A Thesis

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Evan Byrne, B.S.

Graduate Program in Electrical and Computer Engineering

The Ohio State University

2015

Master's Examination Committee:

Dr. Philip Schniter, Advisor

Dr. Lee C. Potter

Abstract

For the problem of multi-class linear classification and feature selection, we propose new approximate message passing algorithms based on Hybrid Generalized Approximate Message Passing (Hybrid-GAMP) to train a multinomial logistic regression model. We propose both maximum a posteriori (MAP) and approximate minimum mean-square error (MMSE) estimators of the weight vectors. Then we design simplified variants of these algorithms that lead to significantly faster runtimes and improved numerical robustness. Our algorithms are able to handle the case where the number of features far exceeds the number of training examples through the use of sparsity-promoting prior distributions on the weight vectors. Additionally, our algorithms are able to take advantage of existing expectation-maximization (EM) and Stein's Unbiased Risk Estimate (SURE) methods to tune their parameters online. Finally, we demonstrate our algorithms' performance on both synthetic and real datasets.

Acknowledgments

Working on this research project was a very fulling experience for me, and I have many people to thank. First, I thank my advisor, Phil Schniter, for his guidance and patience during the course of this project. I thank my advisor again along with Lee Potter and Emre Koksal for teaching fun and intellectually stimulating undergraduate courses. Without these courses I may have never developed an interest in my particular field. Also, I thank everyone involved with the IPS group including Jeri McMichael and the students for being friendly and more than willing to offer help. I also thank my colleagues at previous internships for their encouragement to pursue grad school, and for writing letters of rec at the last second. I thank my friends including my roommates and members of Buckeye Barbell Club for providing much needed distractions for the past couple of years. Lastly, I thank my family for their love and support.

Vita

2012 ..... B.S. Electrical and Computer Engineering, The Ohio State University

2013-present ..... Graduate Research Assistant, The Ohio State University

Fields of Study

Major Field: Electrical and Computer Engineering

Studies in:

Communication Theory	Prof. Philip Schniter
Digital Signal Processing	Prof. Lee Potter

Table of Contents

List of Figures

Figure **Page**

List of Tables

Table **Page**

Chapter 1: Introduction

1.1 Linear classification

In this thesis we explore linear classification and feature selection. In classification [1, pg. 3] we are given training data consisting of M feature-label pairs $\{\boldsymbol{a}_m, y_m\}_{m=1}^M$ where $\boldsymbol{a}_m \in \mathbb{R}^N$ is a feature vector and $y_m \in \{1, ..., D\}$ is a class label. Our objective is to use the training data to predict the unknown class label y_0 on a test feature vector $\boldsymbol{a}_0$.

Linear classification [1, §4] is a two step process. First, we compute linear scores $\boldsymbol{z}_0 \triangleq \boldsymbol{X}^\mathsf{T}\boldsymbol{a}_0 \in \mathbb{R}^D$ using the weight-matrix $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ learned from the training data. Then, we estimate the class label as the index of the largest score:

$$\widehat{y_0} = \arg\max_d [\boldsymbol{z}_0]_d. \tag{1.1}$$

We are interested in classification problems that have the following attributes.

1. They involve $D > 2$ classes.

2. The feature vector dimension N is very large, e.g., $N > 1000$.

3. The number of training examples M is much less than N.

4. A small number, $K \ll \{M, N\}$, of the features may suffice for accurate classification. In this case, the 'true' weight-matrix $\boldsymbol{X}$ is *sparse*.

In the case that a small subset of features suffices for accurate classification, we may want to estimate that subset, which is known as "feature selection." We can do this by simply examining the row-support of $\boldsymbol{X}$.

In practice, we see many applications of multi-class linear classification and feature selection that display our previously listed attributes. For instance, in fMRI Multi-Voxel Pattern Analysis (MVPA), feature selection can be used to determine which areas of the brain are active during particular cognitive tasks [2,3]. In this application, features are indexed over voxels in an fMRI image and their value corresponds to the voxel's brightness, while each class is a particular cognitive task. Another application is text mining, where the objective might be to classify the topic of a document based on the frequency of keywords [4]. In this application, features may be the count or frequency of particular words, while the classes are the article's topic. In bioinformatics, one may want to learn which genes contribute to a particular disease, which can be recognized as feature selection [5,6]. In this application, features may be gene expression levels and classes are particular diseases. Lastly, there are a multitude of other multi-class classification problems, such as handwriting recognition [7], where features are pixels in an image of a character and the classes are the various characters to be recognized. In the previously listed applications, the number of features (N) is typically on the order of 10000, the number of training examples (M) is on the order of 100 to 100000, and the number of classes (D) is on the order of 10.

1.2 Designing the weight matrix

The main challenge in linear classification is designing the weight matrix $\boldsymbol{X}$. We take the approach of assuming that the training data $\{\boldsymbol{a}_m, y_m\}_{m=1}^M$ are independent realizations of the random variable pair (a, y) that obeys an assumed conditional probability model

$$p_{\mathsf{y}\,|\,\mathsf{A}}(\boldsymbol{y}\,|\,\boldsymbol{A};\boldsymbol{X}) = \prod_{m=1}^{M} p_{\mathsf{y}\,|\,\mathsf{a}}(y_m\,|\,\boldsymbol{a}_m;\boldsymbol{X}), \tag{1.2}$$

for $\boldsymbol{y} \triangleq [y_1, ..., y_M]^{\mathsf{T}}$, $\boldsymbol{A} \triangleq [\boldsymbol{a}_1, ..., \boldsymbol{a}_M]^{\mathsf{T}}$ and some "true" weight matrix $\boldsymbol{X}$. Then, one approach is to solve for the weight matrix $\widehat{\boldsymbol{X}}$ that gives us the best fit to the model, i.e.,

$$\widehat{\boldsymbol{X}}_{\mathsf{ML}} = \arg\max_{\boldsymbol{X}} \prod_{m=1}^{M} p_{\mathsf{y}\,|\,\mathsf{a}}(y_m\,|\,\boldsymbol{a}_m;\boldsymbol{X}) = \arg\max_{\boldsymbol{X}} \sum_{m=1}^{M} \log p_{\mathsf{y}\,|\,\mathsf{a}}(y_m\,|\,\boldsymbol{a}_m;\boldsymbol{X}). \tag{1.3}$$

The above can be recognized as the maximum likelihood estimate [8, §IV.D] of $\boldsymbol{X}$, hence the subscript "ML." Our approach of assuming a particular conditional probability model $p_{\mathsf{y}\,|\,\mathsf{A}}(\boldsymbol{y}\,|\,\boldsymbol{A};\boldsymbol{X})$ and finding a weight matrix that best fits the model is known as the "discriminative" approach, in contrast to the "generative" approach, which attempts to find the joint probability $p_{\mathsf{A},\mathsf{y}}(\boldsymbol{A}, \boldsymbol{y})$ directly [1, pg. 43].

In linear classification, the per-example model $p_{\mathsf{y}\,|\,\mathsf{a}}(y_m\,|\,\boldsymbol{a}_m;\boldsymbol{X})$ is based on a linear interaction between $\boldsymbol{a}_m$ and $\boldsymbol{X}$, and thus can be written in terms of the scores $\boldsymbol{z}_m = \boldsymbol{X}^{\mathsf{T}}\boldsymbol{a}_m$. In particular,

$$p_{\mathsf{y}\,|\,\mathsf{a}}(y_m\,|\,\boldsymbol{a}_m;\boldsymbol{X}) = p_{\mathsf{y}\,|\,\mathsf{z}}(y_m\,|\,\boldsymbol{X}^{\mathsf{T}}\boldsymbol{a}_m) \tag{1.4}$$

for the newly introduced random vector $\mathsf{z} \triangleq \boldsymbol{X}^{\mathsf{T}}\mathsf{a}$. The conditional pdf $p_{\mathsf{y}\,|\,\mathsf{z}}(y_m\,|\,\boldsymbol{z}_m)$ is known as the "activation" function [1, pg. 180]. Throughout this thesis, we focus

on the "multinomial logistic" [1, §4.3.4] [9, §2] choice given by

$$p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \boldsymbol{z}_m) = \frac{\exp(z_{y_m})}{\sum_{d=1}^{D} \exp(z_d)}, \quad y \in \{1, ..., D\} \tag{1.5}$$

or equivalently,

$$p_{\mathsf{y}\,|\,\mathsf{a}}(y_m \,|\, \boldsymbol{a}_m; \boldsymbol{X}) = \frac{\exp(\overline{\boldsymbol{x}}_{y_m}^{\mathsf{T}} \boldsymbol{a}_m)}{\sum_{d=1}^{D} \exp(\overline{\boldsymbol{x}}_d^{\mathsf{T}} \boldsymbol{a}_m)}, \quad y \in \{1, ..., D\} \tag{1.6}$$

where $\overline{\boldsymbol{x}}_d$ is the dth *column* of $\boldsymbol{X}$. We will justify using (1.5) in Section 1.3.3.

When the training data is linearly separable, i.e., there exists a weight matrix $\boldsymbol{X}$ that correctly classifies every training example, then $\widehat{\boldsymbol{X}}_{\mathsf{ML}}$ is infinite because from (1.6), $p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \boldsymbol{X}^{\mathsf{T}} \boldsymbol{a}_m)$ is strictly increasing in $\boldsymbol{X}$ for all m. One way to resolve this issue is to place a penalty on large $\boldsymbol{X}$, i.e., to solve for

$$\widehat{\boldsymbol{X}} = \arg\max_{\boldsymbol{X}} \sum_{m=1}^{M} \log p_{\mathsf{y}\,|\,\mathsf{a}}(y_m \,|\, \boldsymbol{a}_m; \boldsymbol{X}) + f(\boldsymbol{X}) \tag{1.7}$$

with appropriately chosen $f(\cdot)$. As we show below, this can be interpreted as maximum a posterior (MAP) estimation of X under the prior $p_{\mathsf{X}}(\boldsymbol{X}) \propto \exp(f(\boldsymbol{X}))$ and the assumption that X is statistically independent of A.

Given the prior pdf $p_{\mathsf{X}}(\boldsymbol{X})$, the MAP [8, §IV.B.3] estimate of X is defined as

$$\widehat{\boldsymbol{X}}_{\mathsf{MAP}} \triangleq \arg\max_{\boldsymbol{X}} p_{\mathsf{X}\,|\,\mathsf{A},\mathsf{y}}(\boldsymbol{X} \,|\, \boldsymbol{A}, \boldsymbol{y}). \tag{1.8}$$

The posterior distribution of X given the training data $\{\boldsymbol{y}, \boldsymbol{A}\}$ is, via Bayes' rule [10],

$$p_{\mathsf{X}\,|\,\mathsf{A},\mathsf{y}}(\boldsymbol{X} \,|\, \boldsymbol{A}, \boldsymbol{y}) = \frac{p_{\mathsf{y}\,|\,\mathsf{A},\mathsf{X}}(\boldsymbol{y} \,|\, \boldsymbol{A}, \boldsymbol{X}) p_{\mathsf{X}\,|\,\mathsf{A}}(\boldsymbol{X} \,|\, \boldsymbol{A})}{p_{\mathsf{y}\,|\,\mathsf{A}}(\boldsymbol{y} \,|\, \boldsymbol{A})} \tag{1.9}$$

$$= \frac{p_{\mathsf{y}\,|\,\mathsf{A},\mathsf{X}}(\boldsymbol{y} \,|\, \boldsymbol{A}, \boldsymbol{X}) p_{\mathsf{X}}(\boldsymbol{X})}{p_{\mathsf{y}\,|\,\mathsf{A}}(\boldsymbol{y} \,|\, \boldsymbol{A})} \tag{1.10}$$

$$\propto p_{\mathsf{y}\,|\,\mathsf{A},\mathsf{X}}(\boldsymbol{y} \,|\, \boldsymbol{A}, \boldsymbol{X}) p_{\mathsf{X}}(\boldsymbol{X}). \tag{1.11}$$

The MAP estimate of $\mathbf{X}$ can thus also be written as

$$\widehat{\boldsymbol{X}}_{\mathsf{MAP}} = \arg\max_{\boldsymbol{X}} \ \log p_{\mathbf{y}\,|\,\mathbf{A},\mathbf{X}}(\boldsymbol{y}\,|\,\boldsymbol{A},\boldsymbol{X}) p_{\mathbf{X}}(\boldsymbol{X}) \tag{1.12}$$

$$= \arg\max_{\boldsymbol{X}} \ \log p_{\mathbf{y}\,|\,\mathbf{A},\mathbf{X}}(\boldsymbol{y}\,|\,\boldsymbol{A},\boldsymbol{X}) + \log p_{\mathbf{X}}(\boldsymbol{X}). \tag{1.13}$$

Note the equivalence between (1.7) and (1.13).

Alternatively, one could seek the minimum mean-squared error (MMSE) estimate [8, §IV.B.1] of $\mathbf{X}$, given by

$$\widehat{\boldsymbol{X}}_{\mathsf{MMSE}} \triangleq \mathrm{E}\{\mathbf{X}\,|\,\mathbf{A},\mathbf{y}\} \tag{1.14}$$

$$= \int_{\boldsymbol{X}\in\mathbb{R}^{N\times D}} \boldsymbol{X} p_{\mathbf{X}\,|\,\mathbf{A},\mathbf{y}}(\boldsymbol{X}\,|\,\boldsymbol{A},\boldsymbol{y})\,\mathrm{d}\boldsymbol{X}. \tag{1.15}$$

However, due to the high-dimensionality of the integral in (1.15), MMSE estimation of $\mathbf{X}$ is typically not done in practice. We will refer to the problems in (1.12) and (1.14), under the assumption of (1.5), collectively as *multinomial logistic regression*.

1.3 Assumed data model

1.3.1 Definition

We now describe a statistical data model that is consistent with multinomial logistic regression. Our data model is defined by a joint pdf $p_{\mathbf{a},\mathsf{y}}(\boldsymbol{a},y)$, where $\mathbf{a}\in\mathbb{R}^N$ is a feature vector and $\mathsf{y}\in\{1,...,D\}$ is its corresponding class label. We prefer to write our model as $p_{\mathbf{a}\,|\,\mathsf{y}}(\boldsymbol{a}\,|\,y)p_{\mathsf{y}}(y)$ to show the dependence of a feature vector on its class label. Our first assumption is uniformly distributed classes, i.e., $p_{\mathsf{y}}(y=d)=\frac{1}{D},\ d\in\{1,...,D\}$. We then assume the feature vector, conditioned on a particular class, has a multivariate Gaussian distribution with a class-dependent mean and class-invariant scaled-identity covariance matrix, i.e.,

$$p_{\mathbf{a}\,|\,\mathsf{y}}(\boldsymbol{a}\,|\,y) = \mathcal{N}(\boldsymbol{a};\boldsymbol{\mu}_y,\sigma_a^2\boldsymbol{I}_N) = \frac{1}{\sqrt{(2\pi\sigma_a^2)^N}}\exp\Big(-\frac{1}{2\sigma_a^2}\|\boldsymbol{a}-\boldsymbol{\mu}_y\|_2^2\Big). \tag{1.16}$$

We furthermore assume the class means have equal norm and are mutually orthogonal, i.e.,

$$\|\boldsymbol{\mu}_d\| = c \;\forall\; d \tag{1.17}$$

and

$$\boldsymbol{\mu}_d^{\mathsf{T}} \boldsymbol{\mu}_{d'} \,\big|_{\, d \neq d'} = 0. \tag{1.18}$$

1.3.2 Bayes' optimal classifier

Given a test feature vector $\boldsymbol{a}_0$, estimating the unknown class y_0 using MAP detection, i.e.,

$$\widehat{y}_0(\boldsymbol{a}_0) = \arg\max_y p_{\mathsf{y}\,|\,\mathbf{a}}(y \,|\, \boldsymbol{a}_0), \tag{1.19}$$

is known to minimize the error probability [8, §II.B]. Using Bayes' rule and our assumption of uniform prior probabilities,

$$p_{\mathsf{y}\,|\,\mathbf{a}}(y \,|\, \boldsymbol{a}_0) = \frac{p_{\mathbf{a}\,|\,\mathsf{y}}(\boldsymbol{a}_0 \,|\, y) p_{\mathsf{y}}(y)}{p_{\mathbf{a}}(\boldsymbol{a}_0)} \tag{1.20}$$

$$= \frac{p_{\mathbf{a}\,|\,\mathsf{y}}(\boldsymbol{a}_0 \,|\, y)}{D p_{\mathbf{a}}(\boldsymbol{a}_0)}, \tag{1.21}$$

and so ML detection is also optimal. Using this, we have

$$\widehat{y}_0(\boldsymbol{a}_0) = \arg\max_y p_{\mathbf{a}\,|\,\mathsf{y}}(\boldsymbol{a}_0 \,|\, y) \tag{1.22}$$

$$= \arg\max_y \mathcal{N}(\boldsymbol{a}_0; \boldsymbol{\mu}_y, \sigma_a^2 I_N) \tag{1.23}$$

$$= \arg\min_y \|\boldsymbol{a}_0 - \boldsymbol{\mu}_y\|_2^2 \tag{1.24}$$

$$= \arg\max_y \boldsymbol{a}_0^{\mathsf{T}} \boldsymbol{\mu}_y, \tag{1.25}$$

where we use $\|\boldsymbol{\mu}_y\| = c \,\forall\, y$ in (1.25). We can see the ML detector uses linear scores $\boldsymbol{a}_0^{\mathsf{T}} \boldsymbol{\mu}_y$ with optimal weight vectors equal to the class means, up to an arbitrary, positive scale factor α, i.e., $\overline{\boldsymbol{x}}_d^* = \alpha \boldsymbol{\mu}_d$ for $d \in \{1, .., D\}$. This problem can be recognized as

standard signal detection in additive white Gaussian noise [8, §III], and in particular, detection of y from measurements

$$\boldsymbol{a} = \boldsymbol{\mu}_y + \boldsymbol{w}, \tag{1.26}$$

where $\boldsymbol{\mu}_y$ is one of D transmitted signals and $\boldsymbol{w}$ is additive noise with distribution $\mathcal{N}(\boldsymbol{w}; \mathbf{0}, \sigma_a^2 I_N)$. The detector structure is known as a "matched filter bank," i.e., (1.25). The minimal (i.e., Bayes') error rate (BER) for this data model (with parameters c and σ_a^2) is

$$\varepsilon_B = 1 - \int \mathcal{N}(z; c/\sigma_a, 1)\Phi(z)^{D-1}\,\mathrm{d}z. \tag{1.27}$$

We derive (1.27) in Appendix A.

1.3.3 Justification for the multinomial logistic likelihood

Next, we show that the model in (1.16) with the assumption of uniform class priors and equal length and mutually orthogonal class means yields the multinomial logistic activation function, given in (1.6). Using Bayes' rule,

$$p_{\mathsf{y}\,|\,\mathbf{a}}(y\,|\,\boldsymbol{a}) = \frac{p_{\mathbf{a}\,|\,\mathsf{y}}(\boldsymbol{a}\,|\,y)p_{\mathsf{y}}(y)}{p_{\mathbf{a}}(\boldsymbol{a})} \tag{1.28}$$

$$= \frac{p_{\mathbf{a}\,|\,\mathsf{y}}(\boldsymbol{a}\,|\,y)p_{\mathsf{y}}(y)}{\sum_{y'=1}^{D} p_{\mathbf{a}\,|\,\mathsf{y}}(\boldsymbol{a}\,|\,y')p_{\mathsf{y}}(y')} \tag{1.29}$$

$$= \frac{1}{1 + \sum_{y'=1,\neq y}^{D} \frac{\mathcal{N}(\boldsymbol{a};\boldsymbol{\mu}_{y'},\sigma_a^2\boldsymbol{I}_N)}{\mathcal{N}(\boldsymbol{a};\boldsymbol{\mu}_y,\sigma_a^2\boldsymbol{I}_N)}} \tag{1.30}$$

$$= \frac{1}{1 + \sum_{y'=1,\neq y}^{D} \exp\left(\frac{1}{\sigma_a^2}\boldsymbol{a}^{\mathsf{T}}(\boldsymbol{\mu}_{y'} - \boldsymbol{\mu}_y)\right)} \tag{1.31}$$

$$= \frac{\exp\left(\frac{1}{\sigma_a^2}\boldsymbol{a}^{\mathsf{T}}\boldsymbol{\mu}_y\right)}{\sum_{y'=1}^{D} \exp\left(\frac{1}{\sigma_a^2}\boldsymbol{a}^{\mathsf{T}}\boldsymbol{\mu}_{y'}\right)}, \tag{1.32}$$

where (1.29) is an application of the law of total probability [10], (1.30) simplifies due to uniform priors, (1.31) results due to $\|\boldsymbol{\mu}_y\| = \|\boldsymbol{\mu}_{y'}\|$ and (1.32) is a restructuring of (1.31). Note that (1.32) matches (1.6) when $\frac{1}{\sigma_a^2}\boldsymbol{\mu}_d = \overline{\boldsymbol{x}}_d$ for all d.

The derivation (1.28)-(1.32) actually did not require the class means to be orthogonal. However, we keep the orthogonality assumption because it was necessary for the probability of error expression (1.27). We note that the multinomial logistic activation function is also optimal under other data models, as described in [1, §4.2].

1.4 Application of approximate message passing algorithms

Recently, an efficient algorithm has been proposed to solve high-dimensional inference problems similar to, but not exactly the same, as those in (1.12) and (1.14). It goes by the name of generalized approximate message passing (GAMP) [11]. In particular, GAMP tries to estimate the random vector $\mathsf{x} \in \mathbb{R}^N$ with separable prior pdf $p_{\mathsf{x}}(\boldsymbol{x}) = \prod_{n=1}^N p_{\mathsf{x}}(x_n)$ from a degraded copy $\boldsymbol{y} \in \mathbb{R}^M$ of the linear transform output $\mathsf{z} = \boldsymbol{A}\mathsf{x} \in \mathbb{R}^M$. Here, the likelihood function of z is assumed to also be separable, i.e., $p_{\mathsf{y}\,|\,\mathsf{z}} = \prod_{m=1}^M p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, z_m)$ and $\boldsymbol{A}$ is a known realization of a large, random matrix.

GAMP computes its estimate $\widehat{\boldsymbol{x}}$ through approximate-MMSE or MAP estimation by approximating the sum-product [1, §8.4.4] or max-sum [1, §8.4.5] algorithms, respectively, on the loopy factor graph [1, §8.4] corresponding to its assumed model.

We cannot apply GAMP to our multinomial logistic regression problem because GAMP assumes a scalar-input/scalar-output channel $p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, z_m)$. However, a more general version of GAMP, known as Hybrid-GAMP (or HyGAMP) [12] has been recently developed and can be applied to our problem, as we show in Chapter 2.

1.5 Prior work

Several sparsity promoting multinomial logistic regression algorithms have been proposed (e.g., [9, 13–17]), differing in their choice of $p_{\mathbf{x}}$ and methodology used to estimate $\mathbf{X}$. For example, [9, 14, 15] use the i.i.d Laplacian prior

$$p_{\mathbf{x}}(\boldsymbol{x}_n; \lambda) = \prod_{d=1}^{D} \frac{\lambda}{2} \exp(-\lambda |x_{nd}|), \tag{1.33}$$

with λ tuned via cross-validation. To circumvent this tuning problem, [16] employs the Laplacian scale mixture

$$p_{\mathbf{x}}(\boldsymbol{x}_n) = \prod_{d=1}^{D} \int \left[\frac{\lambda}{2} \exp(-\lambda |x_{nd}|) \right] p(\lambda) \, \mathrm{d}\lambda, \tag{1.34}$$

with Jeffrey's non-informative hyperprior $p(\lambda) \propto \frac{1}{\lambda} 1_{\lambda \geq 0}$. The relevance vector machine (RVM) approach [13] uses the Gaussian scale mixture

$$p_{\mathbf{x}}(\boldsymbol{x}_n) = \prod_{d=1}^{D} \int \mathcal{N}(x_{nd}; 0, \nu) p(\nu) \, \mathrm{d}\nu, \tag{1.35}$$

with inverse-gamma $p(\nu)$ (i.e., the conjugate hyperprior), resulting in an i.i.d. student's t distribution for $p_{\mathbf{x}}$. 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 (1.33) for $p_{\mathbf{x}}$ [18]. Finally, [17] uses

$$p_{\mathbf{x}}(\boldsymbol{x}_n; \lambda) \propto \exp(-\lambda \|\boldsymbol{x}_n\|_2), \tag{1.36}$$

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

Once the probabilistic model (1.9) has been specified, a procedure is needed to infer the weights $\mathbf{X}$ from the training data $\{(y_m, \boldsymbol{a}_m)\}_{m=1}^M$. The Laplacian-prior methods [9, 14, 15, 17] use the maximum a posteriori (MAP) estimation framework:

$$\widehat{\boldsymbol{X}} = \arg\max_{\boldsymbol{X}} \log p(\boldsymbol{X}|\boldsymbol{y}; \boldsymbol{A}) \tag{1.37}$$

$$= \arg\max_{\boldsymbol{X}} \sum_{m=1}^{M} \log p_{\mathsf{y}|\mathsf{z}}(y_m|\boldsymbol{X}^{\mathsf{T}}\boldsymbol{a}_m) + \sum_{n=1}^{N} \log p_{\mathbf{x}}(\boldsymbol{x}_n), \tag{1.38}$$

where Bayes rule was used for (1.38). Under $p_{\mathbf{x}}$ from (1.33) or (1.36), the second term in (1.38) 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, (1.38) is concave and can be maximized in polynomial time; [9, 14, 15, 17] employ (block) coordinate ascent for this purpose. The papers [13] and [16] handle the scale-mixture priors (1.34) and (1.35), respectively, using the evidence maximization framework [19]. This approach yields a double-loop procedure: the hyperparameter λ and ν is estimated in the outer loop, and—for fixed λ and ν—the resulting concave (i.e., ℓ_2 and ℓ_1 regularized) MAP optimization is solved in the inner loop.

The methods [9, 13–17] 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 (1.1).

Finally, GAMP has previously been applied to binary linear classification in [20], i.e., the special case of our problem where $D = 2$.

1.6 Contributions

In this thesis, we first develop two algorithms for multinomial logistic regression by applying Hybrid-GAMP to both (1.12) and (1.14). We will refer to these two algorithms as MAP-HyGAMP and MMSE-HyGAMP, respectively. Then, we simplify

both of these algorithms to obtain simplified Hybrid-GAMP i.e., MAP-SHyGAMP and MMSE-SHyGAMP. Finally, through a numerical study we demonstrate that MAP-SHyGAMP and MMSE-SHyGAMP are competitive with state-of-the-art sparse multinomial logistic regression algorithms.

1.7 Thesis outline

In Chapter 2, we apply Hybrid-GAMP to the MAP (1.12) and MMSE (1.14) formulations of the multinomial logistic regression problem. In Chapter 3, we simplify Hybrid-GAMP to form two new algorithms that exhibit faster runtimes. In Chapter 4, we demonstrate our algorithms' state-of-the-art performance on both synthetic and real data. Finally, in Chapter 5, we present a brief summary of our algorithms and a description of their performance.

Notation We represent matrices with boldface uppercase letters, such as $\boldsymbol{A}$, and we represent vectors with boldface lowercase letters, such as $\boldsymbol{x}$. Scalar quantities will not be boldface. Random quantities will be denoted with sans-serif font (e.g., x, $\mathbf{x}$, $\mathbf{X}$), while deterministic quantities will be denoted with serif font (e.g., x, $\boldsymbol{x}$, $\boldsymbol{X}$). Unless otherwise stated, subscripts are used to index elements, rows or columns in a matrix or vector, e.g., z_d is the dth element in $\boldsymbol{z}$. Similarly, x_{nd} is the dth element in the vector $\boldsymbol{x}_n$. Probability distributions will have their subscripts abbreviated when the random variables in question are i.i.d., e.g., $p_{\mathsf{x}_n}(x_n) = p_{\mathsf{x}}(x_n)$. If $\boldsymbol{x}$ is normally distributed with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ this will be denoted by $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$. $\mathrm{var}\{\mathbf{x}\}$ indicates the covariance matrix of $\mathbf{x}$. $\frac{\partial^2}{\partial \boldsymbol{x}^2} f(\boldsymbol{x})$ indicates the Hessian of $f(\boldsymbol{x})$. Also, let $\phi(x)$ and $\Phi(x)$ represent the standard normal pdf and cdf, respectively, evaluated

at x. Finally, Table 1.1 provides a summary of important variables used throughout this thesis.

Variable	Definition
N	feature vector dimension
M	number of training examples
D	number of classes
K	'true' weight-matrix row-sparsity
$\boldsymbol{a}_m \in \mathbb{R}^N$	feature vector
$\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_M]^\mathsf{T} \in \mathbb{R}^{M \times N}$	feature matrix
$y_m \in \{1, ..., D\}$	class label corresponding to $\boldsymbol{a}_m$
$\boldsymbol{X} \in \mathbb{R}^{N \times D}$	weight matrix
$\boldsymbol{z}_m = \boldsymbol{X}^\mathsf{T} \boldsymbol{a}_m \in \mathbb{R}^D$	scores corresponding to $\boldsymbol{a}_m$

Table 1.1: Variable definitions

Chapter 2: Hybrid-GAMP for Multinomial Logistic Regression

In this chapter we apply Hybrid-GAMP to the MAP (1.12) and MMSE (1.14) formulations of the multinomial logistic regression problem, yielding the algorithms MAP-HyGAMP and MMSE-HyGAMP respectively.

2.1 Background on Hybrid-GAMP

Hybrid-GAMP [12] is a recently developed extension of the GAMP algorithm [11] that performs high-dimensional inference with more general statistical models. One such model, which is markedly similar to the GAMP model described in Section 1.4, is the following: estimate $\mathsf{X} \in \mathbb{R}^{N\times D}$ with separable prior pdf $p_{\mathsf{X}}(\boldsymbol{X}) = \prod_{n=1}^{N} p_{\mathsf{x}}(\boldsymbol{x}_n)$ (where $\boldsymbol{x}_n^{\mathsf{T}}$ is the nth row of $\boldsymbol{X}$) from noisy observations $\boldsymbol{y} \in \mathbb{R}^M$ of the linear transform output $\mathsf{Z} = \boldsymbol{A}\mathsf{X} \in \mathbb{R}^{M\times D}$. Here, the likelihood function of Z is assumed to be separable, i.e., $p_{\mathsf{y}\,|\,\mathsf{Z}}(\boldsymbol{y}\,|\,\boldsymbol{Z}) = \prod_{m=1}^{M} p_{\mathsf{y}\,|\,\mathsf{z}}(y_m\,|\,\boldsymbol{z}_m)$ (where $\boldsymbol{z}_m^{\mathsf{T}}$ is the mth row of $\boldsymbol{Z}$), and $\boldsymbol{A}$ is assumed to be a known realization of a large, random matrix.

The posterior density for this particular statistical model can be written as

$$p_{\mathsf{X}\,|\,\mathsf{y}}(\boldsymbol{X}\,|\,\boldsymbol{y}) \propto p_{\mathsf{y}\,|\,\mathsf{Z}}(\boldsymbol{y}\,|\,\boldsymbol{A}\boldsymbol{X}) p_{\mathsf{X}}(\boldsymbol{X}) \tag{2.1}$$

$$\propto \prod_{m=1}^{M} p_{\mathsf{y}\,|\,\mathsf{z}}(y_m\,|\,\boldsymbol{a}_m^{\mathsf{T}}\boldsymbol{X}) \prod_{n=1}^{N} p_{\mathsf{x}}(\boldsymbol{x}_n), \tag{2.2}$$

where $\boldsymbol{a}_m^{\mathsf{T}}$ is the mth row of $\boldsymbol{A}$. The pdf in (2.2) can be represented with a bipartite factor graph, shown in Figure 2.1. Hybrid-GAMP finds a MAP or approximate-

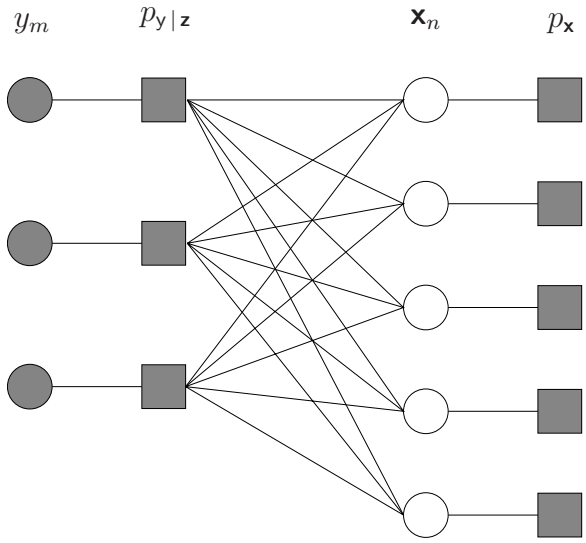

Figure 2.1: Factor graph corresponding to (2.2). Square nodes represent functions while circular nodes represent variables. Shaded nodes are known, while non-shaded nodes are unknown. The nodes on the left hand side represent the likelihood and the nodes on the right hand side represent the prior. Nodes are *vector*-valued where appropriate in accordance with the Hybrid-GAMP model.

MMSE estimate of $\mathbf{X}$ by approximating the max-sum or sum-product algorithm, respectively. Through message passing, Hybrid-GAMP breaks this high dimensional inference problem into many lower-dimensional inference problems (of size D). Hybrid-GAMP iteratively sends messages back and forth between the nodes on the left and right side of the graph in Figure 2.1. These messages take the form of multivariate Gaussian pdfs. The Gaussianity of the messages follows in the large system limit from the central limit theorem [10].

Hybrid-GAMP approximates the marginal posterior distribution of the weights x_n with a distribution of the form

$$p_{\mathsf{x}|\mathsf{r}}(\boldsymbol{x}_n \,|\, \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}_n^{\mathsf{r}}) \propto p_{\mathsf{x}}(\boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}_n^{\mathsf{r}}), \tag{2.3}$$

where $\widehat{\boldsymbol{r}}_n$ and $\boldsymbol{Q}_n^{\mathsf{r}}$ are computed iteratively. Likewise, Hybrid-GAMP approximates the marginal posterior distribution of the transformed weights z_m with

$$p_{\mathsf{z}|\mathsf{y},\mathsf{p}}(\boldsymbol{z}_m \,|\, y_m, \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}_m^{\mathsf{p}}) \propto p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}_m^{\mathsf{p}}), \tag{2.4}$$

where $\widehat{\boldsymbol{p}}_m$ and $\boldsymbol{Q}_m^{\mathsf{p}}$ are computed iteratively.

The Hybrid-GAMP algorithm is given in Algorithm 1. Note that the algorithm is broken into linear and non-linear steps; the linear steps are the same for both MAP-HyGAMP and MMSE-HyGAMP, but the non-linear steps (also referred to as input and output estimators) differ based on the estimator (MAP or MMSE) used. Further note that each iteration of the HyGAMP algorithm requires $M + N$ $D \times D$ matrix inverses. The input estimators involve finding an estimate for x_n using the approximate posterior distribution in (2.3) and the output estimators involve finding an estimate for z_m using the approximate posterior distribution in (2.4). The input and output estimation steps match the overall theme of the algorithm: in MAP-HyGAMP a small MAP estimation problem is solved at each node, likewise in MMSE-HyGAMP a small MMSE estimation problem is solved at each node.

2.2 Hybrid-GAMP for multinomial logistic regression

Consider our problem of multi-class linear classification via multinomial logistic regression. We have M training examples $\{y_m, \boldsymbol{a}_m\}_{m=1}^M$, where $y_m \in \{1, ..., D\}$ and $\boldsymbol{a}_m \in \mathbb{R}^N$. We want to design the weight matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ where each column of

Algorithm 1 HyGAMP

Require: Mode $\in \{\texttt{MAP}, \texttt{MMSE}\}$, matrix $\boldsymbol{A}$, vector $\boldsymbol{y}$, pdfs $p_{\mathbf{x}|\mathbf{r}}$ and $p_{\mathbf{z}|\mathsf{y},\mathbf{p}}$ from (2.3)-(2.4), initializations $\widehat{\boldsymbol{r}}_n(0)$, $\boldsymbol{Q}^{\mathbf{r}}_n(0)$.

Ensure: $t \leftarrow 0$; $\widehat{\boldsymbol{s}}_m(0) \leftarrow \mathbf{0}$.

repeat
  if MAP **then** {**for** $n = 1 \dots N$}
    $\widehat{\boldsymbol{x}}_n(t) \leftarrow \arg\max_{\boldsymbol{x}} \log p_{\mathbf{x}|\mathbf{r}}\big(\boldsymbol{x}_n \big| \widehat{\boldsymbol{r}}_n(t-1); \boldsymbol{Q}^{\mathbf{r}}_n(t-1)\big)$
    $\boldsymbol{Q}^{\mathbf{x}}_n(t) \leftarrow \big[-\frac{\partial^2}{\partial \boldsymbol{x}^2} \log p_{\mathbf{x}|\mathbf{r}}\big(\widehat{\boldsymbol{x}}_n(t) \big| \widehat{\boldsymbol{r}}_n(t-1); \boldsymbol{Q}^{\mathbf{r}}_n(t-1)\big)\big]^{-1}$
  else if MMSE **then** {**for** $n = 1 \dots N$}
    $\widehat{\boldsymbol{x}}_n(t) \leftarrow \mathrm{E}\left\{\mathbf{x}_n \,\middle|\, \mathbf{r}_n = \widehat{\boldsymbol{r}}_n(t-1); \boldsymbol{Q}^{\mathbf{r}}_n(t-1)\right\}$
    $\boldsymbol{Q}^{\mathbf{x}}_n(t) \leftarrow \mathrm{Cov}\left\{\mathbf{x}_n \,\middle|\, \mathbf{r}_n = \widehat{\boldsymbol{r}}_n(t-1); \boldsymbol{Q}^{\mathbf{r}}_n(t-1)\right\}$
  end if
  $\forall m:\ \boldsymbol{Q}^{\mathbf{p}}_m(t) \leftarrow \sum_{n=1}^N A_{mn}^2 \boldsymbol{Q}^{\mathbf{x}}_n(t)$
  $\forall m:\ \widehat{\boldsymbol{p}}_m(t) \leftarrow \sum_{n=1}^N A_{mn} \widehat{\boldsymbol{x}}_n(t) - \boldsymbol{Q}^{\mathbf{p}}_m(t) \widehat{\boldsymbol{s}}_m(t-1)$
  if MAP **then** {**for** $m = 1 \dots M$}
    $\widehat{\boldsymbol{z}}_m(t) \leftarrow \arg\max_{\boldsymbol{z}} \log p_{\mathbf{z}|\mathsf{y},\mathbf{p}}\big(\boldsymbol{z}_m \big| y_m, \widehat{\boldsymbol{p}}_m(t); \boldsymbol{Q}^{\mathbf{p}}_m(t)\big)$
    $\boldsymbol{Q}^{\mathbf{z}}_m(t) \leftarrow \big[-\frac{\partial^2}{\partial \boldsymbol{z}^2} \log p_{\mathbf{z}|\mathsf{y},\mathbf{p}}\big(\widehat{\boldsymbol{z}}_m(t) \big| y_m, \widehat{\boldsymbol{p}}_m(t); \boldsymbol{Q}^{\mathbf{p}}_m(t)\big)\big]^{-1}$
  else if MMSE **then** {**for** $m = 1 \dots M$}
    $\widehat{\boldsymbol{z}}_m(t) \leftarrow \mathrm{E}\left\{\mathbf{z}_m \,\middle|\, y_m, \mathbf{p}_m = \widehat{\boldsymbol{p}}_m(t); \boldsymbol{Q}^{\mathbf{p}}_m(t)\right\}$
    $\boldsymbol{Q}^{\mathbf{z}}_m(t) \leftarrow \mathrm{Cov}\left\{\mathbf{z}_m \,\middle|\, y_m, \mathbf{p}_m = \widehat{\boldsymbol{p}}_m(t); \boldsymbol{Q}^{\mathbf{p}}_m(t)\right\}$
  end if
  $\forall m:\ \boldsymbol{Q}^{\mathbf{s}}_m(t) \leftarrow [\boldsymbol{Q}^{\mathbf{p}}_m(t)]^{-1} - [\boldsymbol{Q}^{\mathbf{p}}_m(t)]^{-1} \boldsymbol{Q}^{\mathbf{z}}_m(t) [\boldsymbol{Q}^{\mathbf{p}}_m(t)]^{-1}$
  $\forall m:\ \widehat{\boldsymbol{s}}_m(t) \leftarrow [\boldsymbol{Q}^{\mathbf{p}}_m(t)]^{-1} \big(\widehat{\boldsymbol{z}}_m(t) - \widehat{\boldsymbol{p}}_m(t)\big)$
  $\forall n:\ \boldsymbol{Q}^{\mathbf{r}}_n(t) \leftarrow \big[\sum_{m=1}^M A_{mn}^2 \boldsymbol{Q}^{\mathbf{s}}_m(t)\big]^{-1}$
  $\forall n:\ \widehat{\boldsymbol{r}}_n(t) \leftarrow \widehat{\boldsymbol{x}}_n(t) + \boldsymbol{Q}^{\mathbf{r}}_n(t) \sum_{m=1}^M A_{mn} \widehat{\boldsymbol{s}}_m(t)$
  $t \leftarrow t + 1$
until Terminated

$\boldsymbol{X}$ is a weight vector. Given $\boldsymbol{z}_m^\mathsf{T} = \boldsymbol{a}_m^\mathsf{T}\boldsymbol{X} \in \mathbb{R}^D$, we model the likelihood of $\boldsymbol{a}_m$ being in class y with the multinomial logistic activation function, i.e.,

$$p_{\mathsf{y}\,|\,\mathsf{z}}(y\,|\,\boldsymbol{z}) = \frac{\exp(z_y)}{\sum_{d=1}^{D}\exp(z_d)}, \quad y \in \{1,...,D\}, \tag{2.5}$$

where z_d is the dth element in $\boldsymbol{z}$ (with the m subscript dropped temporarily for brevity). By assuming independent training examples and also prior independence among the *rows* of $\mathbf{X}$, we have the following posterior distribution on the weight matrix given the training data

$$p_{\mathbf{X}\,|\,\mathsf{y}}(\boldsymbol{X}\,|\,\boldsymbol{y}) \propto p_{\mathsf{y}\,|\,\mathsf{Z}}(\boldsymbol{y}\,|\,\boldsymbol{A}\boldsymbol{X})p_{\mathbf{X}}(\boldsymbol{X}) \tag{2.6}$$

$$\propto \prod_{m=1}^{M} p_{\mathsf{y}\,|\,\mathsf{z}}(y_m\,|\,\boldsymbol{a}_m^\mathsf{T}\boldsymbol{X}) \prod_{n=1}^{N} p_{\mathbf{x}}(\boldsymbol{x}_n), \tag{2.7}$$

where $\boldsymbol{A} = [\boldsymbol{a}_1,...,\boldsymbol{a}_M]^\mathsf{T}$ and $\boldsymbol{x}_n^\mathsf{T} \in \mathbb{R}^D$ is the nth row of $\boldsymbol{X}$. Our model in (2.7) exactly matches that assumed by Hybrid-GAMP, given in (2.2).

2.2.1 Multinomial logistic regression via MAP-HyGAMP

In order to perform multinomial logistic regression with MAP-HyGAMP, we must compute line 3 of Algorithm 1, the MAP estimate of $\mathbf{x}_n$, at each input node n using the approximate posterior distribution given in (2.3). Likewise, we must compute line 12 of Algorithm 1, the MAP estimate of $\mathbf{z}_m$ at each output node m using the approximate posterior distribution given in (2.4), where $p_{\mathsf{y}\,|\,\mathsf{z}}$ is the multinomial logistic activation function given in (2.5). In addition, we must compute the inverse Hessian of the log approximate posterior at the input and the output, shown in lines 4 and 13 of Algorithm 1, evaluated at the corresponding MAP estimates, $\widehat{\boldsymbol{x}}_n$ and $\widehat{\boldsymbol{z}}_m$, respectively.

Input estimators: inference of $\widehat{\boldsymbol{x}}_n$

In MAP-HyGAMP, the computation required in line 3 of Algorithm 1, given (2.3), is

$$\widehat{\boldsymbol{x}}_n = \underset{\boldsymbol{x}_n \in \mathbb{R}^D}{\arg\max} \left\{ \log p_{\mathsf{x}\,|\,\mathsf{r}}(\boldsymbol{x}_n \,|\, \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}^{\mathsf{r}}_n) \right\} \tag{2.8}$$

$$= \underset{\boldsymbol{x}_n \in \mathbb{R}^D}{\arg\max} \left\{ \log p_{\mathsf{x}}(\boldsymbol{x}_n) - \frac{1}{2}(\boldsymbol{x}_n - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}} [\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} (\boldsymbol{x}_n - \widehat{\boldsymbol{r}}_n) \right\} \tag{2.9}$$

and the computation required in line 4 is

$$\boldsymbol{Q}^{\mathsf{x}}_n = \left[-\frac{\partial^2}{\partial \boldsymbol{x}_n^2} \log p_{\mathsf{x}\,|\,\mathsf{r}}(\widehat{\boldsymbol{x}}_n \,|\, \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}^{\mathsf{r}}_n) \right]^{-1}, \tag{2.10}$$

with the iteration index t omitted here for simplicity. Note that in (2.10), the Hessian of $\log p_{\mathsf{x}\,|\,\mathsf{r}}(\cdot \,|\, \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}^{\mathsf{r}}_n)$ is evaluated at the $\widehat{\boldsymbol{x}}_n$ found in (2.8).

Choice of prior As discussed in Section 1.2, we are motivated to use a separable, log-concave prior on $\mathbf{X}$ in order to guarantee a finite $\widehat{\boldsymbol{X}}$ and possibly to promote sparsity. Two such choices for the prior are the Gaussian or Laplacian distribution [9].

Gaussian Under a Gaussian prior on x_n defined by $p_{\mathsf{x}}(\boldsymbol{x}_n) = \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}^x_0, \boldsymbol{\Sigma}^x_0)$, the posterior distribution of x_n is a Gaussian and thus it obtains its maximum at its mean. This results in the following closed-form solution for (2.9):

$$\widehat{\boldsymbol{x}}_n = ([\boldsymbol{\Sigma}^x_0]^{-1} + [\boldsymbol{Q}^{\mathsf{r}}_n]^{-1})^{-1} ([\boldsymbol{\Sigma}^x_0]^{-1} \boldsymbol{\mu}^x_0 + [\boldsymbol{Q}^{\mathsf{r}}_n]^{-1} \widehat{\boldsymbol{r}}_n). \tag{2.11}$$

Equation (2.11) follows from the following Gaussian multiplication rule:

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{a}; \boldsymbol{A}) \mathcal{N}(\boldsymbol{x}; \boldsymbol{b}, \boldsymbol{B}) = \mathcal{N}(\boldsymbol{0}; \boldsymbol{a} - \boldsymbol{b}, \boldsymbol{A} + \boldsymbol{B}) \mathcal{N}\Big(\boldsymbol{x}; (\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1} (\boldsymbol{A}^{-1}\boldsymbol{a} + \boldsymbol{B}^{-1}\boldsymbol{b}), (\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1}\Big). \tag{2.12}$$

A Gaussian prior imposes an ℓ_2 norm penalty on $\widehat{\boldsymbol{x}}_n$, which is *not* sparsity-promoting. Additionally, (2.10) can also be found in closed-form via

$$\boldsymbol{Q}_n^{\mathsf{x}} = ([\boldsymbol{\Sigma}_0^x]^{-1} + [\boldsymbol{Q}_n^{\mathsf{r}}]^{-1})^{-1}. \tag{2.13}$$

Laplacian A Laplacian prior,

$$p_{\boldsymbol{x}}(\boldsymbol{x}_n) \propto \exp(-\lambda\|\boldsymbol{x}\|_1), \tag{2.14}$$

imposes an ℓ_1 norm penalty on $\widehat{\boldsymbol{x}}_n$, which *is* sparsity-promoting. In this case, (2.9) becomes

$$\widehat{\boldsymbol{x}}_n = \underset{\boldsymbol{x}_n \in \mathbb{R}^D}{\arg\max} -\lambda\|\boldsymbol{x}_n\|_1 - \frac{1}{2}(\boldsymbol{x}_n - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}}[\boldsymbol{Q}_n^{\mathsf{r}}]^{-1}(\boldsymbol{x}_n - \widehat{\boldsymbol{r}}_n), \tag{2.15}$$

which is reminiscent of the LASSO [21] problem. Although (2.15) has no closed-form solution, it can be solved iteratively using minorize-maximization (MM) [22].

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

$$\widehat{\boldsymbol{x}}^{(k+1)} = \arg\max_{\boldsymbol{x}} \widehat{J}(\boldsymbol{x}; \widehat{\boldsymbol{x}}^{(k)}), \tag{2.16}$$

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 (2.15), we identify the utility function as $J_n(\boldsymbol{x}) \triangleq -\frac{1}{2}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}}[\boldsymbol{Q}_n^{\mathsf{r}}]^{-1}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n) - \lambda\|\boldsymbol{x}\|_1$. Next we apply a result from [23] that established that $J_n(\boldsymbol{x})$ is minorized by $\widehat{J}_n(\boldsymbol{x}; \widehat{\boldsymbol{x}}_n^{(k)}) \triangleq -\frac{1}{2}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}}[\boldsymbol{Q}_n^{\mathsf{r}}]^{-1}(\boldsymbol{x} - \widehat{\boldsymbol{r}}_n) - \frac{\lambda}{2}\big(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}_n^{(k)})\boldsymbol{x} + \|\widehat{\boldsymbol{x}}_n^{(k)}\|_2^2\big)$ with $\boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}) \triangleq \operatorname{diag}\big\{|\widehat{x}_1|^{-1}, \ldots, |\widehat{x}_D|^{-1}\big\}$. Thus (2.16) implies

$$\widehat{\boldsymbol{x}}_n^{(k+1)} = \arg\max_{\boldsymbol{x}} \widehat{J}_n(\boldsymbol{x}; \widehat{\boldsymbol{x}}_n^{(k)}) \tag{2.17}$$

$$= \arg\max_{\boldsymbol{x}} \boldsymbol{x}^{\mathsf{T}}[\boldsymbol{Q}_n^{\mathsf{r}}]^{-1}\widehat{\boldsymbol{r}}_n - \frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\big([\boldsymbol{Q}_n^{\mathsf{r}}]^{-1} + \lambda\boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}_n^{(k)})\big)\boldsymbol{x} \tag{2.18}$$

$$= \big([\boldsymbol{Q}_n^{\mathsf{r}}]^{-1} + \lambda\boldsymbol{\Lambda}(\widehat{\boldsymbol{x}}_n^{(k)})\big)^{-1}[\boldsymbol{Q}_n^{\mathsf{r}}]^{-1}\widehat{\boldsymbol{r}}_n \tag{2.19}$$

where (2.18) dropped the $\boldsymbol{x}$-invariant terms from $\widehat{J}_n(\boldsymbol{x}; \widehat{\boldsymbol{x}}_n^{(k)})$. Note that each iteration k of (2.19) requires a $D \times D$ matrix inverse.

The Hessian of the log of (2.14) does not exist when $x_{nd} = 0$ for any d and is zero otherwise. Thus, we set $\boldsymbol{Q}_n^{\mathsf{x}} = \boldsymbol{Q}_n^{\mathsf{r}}$, but then zero the dth row and column of $\boldsymbol{Q}_n^{\mathsf{x}}$ for all d such that $\widehat{x}_{nd} = 0$.

Output estimators: inference of $\widehat{\boldsymbol{z}}_m$

In MAP-HyGAMP, the computation required at line 12 of Algorithm 1, given (2.4), is

$$\widehat{\boldsymbol{z}}_m = \underset{\boldsymbol{z}_m \in \mathbb{R}^D}{\arg\max} \left\{ \log p_{\mathsf{z}\,|\,\mathsf{y},\mathsf{p}}(\boldsymbol{z}_m \,|\, y_m, \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}_m^{\mathsf{p}}) \right\} \tag{2.20}$$

$$= \underset{\boldsymbol{z}_m \in \mathbb{R}^D}{\arg\max} \left\{ \log p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \boldsymbol{z}_m) - \frac{1}{2}(\boldsymbol{z}_m - \widehat{\boldsymbol{p}}_m)^{\mathsf{T}} [\boldsymbol{Q}_m^{\mathsf{p}}]^{-1} (\boldsymbol{z}_m - \widehat{\boldsymbol{p}}_m) \right\} \tag{2.21}$$

and the computation required at line 13 is

$$\boldsymbol{Q}_m^{\mathsf{z}} = \left[-\frac{\partial^2}{\partial \boldsymbol{z}_m^2} \left\{ \log p_{\mathsf{z}\,|\,\mathsf{y},\mathsf{p}}(\widehat{\boldsymbol{z}}_m \,|\, y_m, \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}_m^{\mathsf{p}}) \right\} \right]^{-1}, \tag{2.22}$$

with the iteration index t again omitted here for simplicity. Similar to (2.10), note that in (2.22) the Hessian of $\log p_{\mathsf{z}\,|\,\mathsf{y},\mathsf{p}}(\cdot \,|\, y_m, \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}_m^{\mathsf{p}})$ is evaluated at the $\widehat{\boldsymbol{z}}_m$ computed in (2.21).

Newton's method We can solve (2.21) using Newton's method [24, §1.4], which uses the iteration

$$\boldsymbol{z}_m^{(k+1)} = \boldsymbol{z}_m^{(k)} + \alpha^{(k)} [\boldsymbol{H}(\boldsymbol{z}_m^{(k)})]^{-1} \nabla(\boldsymbol{z}_m^{(k)}), \tag{2.23}$$

where $\nabla(\cdot)$ and $\boldsymbol{H}(\cdot)$ are the gradient and Hessian, respectively, of the objective function in (2.21) and $\alpha^{(k)}$ is a positive step size.

Given (2.5), the gradient of $\log p_{\mathsf{y}\,|\,\mathbf{z}}(y_m \,|\, \boldsymbol{z}_m)$ w.r.t. $\boldsymbol{z}_m$ is

$$\frac{\partial}{\partial \boldsymbol{z}} \log p_{\mathsf{y}\,|\,\mathbf{z}}(y_m \,|\, \boldsymbol{z}_m) = \frac{\partial}{\partial \boldsymbol{z}} \left(z_{m,y_m} - \log \left(\sum_{d=1}^{D} \exp(z_{m,d}) \right) \right) \tag{2.24}$$

$$= \boldsymbol{y}'_m - \boldsymbol{s}(\boldsymbol{z}_m), \tag{2.25}$$

where $\boldsymbol{y}'_m$ is a length-D vector containing a 1 in the y_mth position and zeros elsewhere, and $\boldsymbol{s}(\boldsymbol{z}_m)$ is a length-D vector where the dth position indicates the value of (2.5) evaluated at $y = d$ and $\boldsymbol{z} = \boldsymbol{z}_m$. The gradient of the objective function in (2.21) is then given by

$$\nabla(\boldsymbol{z}_m) = \boldsymbol{y}'_m - \boldsymbol{s}(\boldsymbol{z}_m) - [\boldsymbol{Q}^{\mathbf{p}}_m]^{-1}(\boldsymbol{z}_m - \widehat{\boldsymbol{p}}_m). \tag{2.26}$$

Given (2.25), the Hessian of $\log p_{\mathsf{y}\,|\,\mathbf{z}}(y_m \,|\, \boldsymbol{z}_m)$ is invariant to y_m (due to $\boldsymbol{y}'_m$ being a constant), so we drop the m subscript for brevity. The Hessian of $\log p_{\mathsf{y}\,|\,\mathbf{z}}(y \,|\, \boldsymbol{z})$ is then

$$\frac{\partial^2}{\partial z_i \partial z_j} \log p_{\mathsf{y}\,|\,\mathbf{z}}(y \,|\, \boldsymbol{z}) = \frac{\partial}{\partial z_j} \left(I_{\{i\}}(y) - \frac{\exp(z_i)}{\sum_{d=1}^{D} \exp(z_d)} \right) \tag{2.27}$$

$$= \frac{-\exp(z_i)\big(\sum_{d=1}^{D} \exp(z_d)\big) + \exp(z_i)^2}{\big(\sum_{d=1}^{D} \exp(z_d)\big)^2} \text{ if } i = j \tag{2.28}$$

$$= \frac{\exp(z_i)\exp(z_j)}{\big(\sum_{d=1}^{D} \exp(z_d)\big)^2} \text{ if } i \neq j, \tag{2.29}$$

which can be written as $-\mathrm{diag}\{\boldsymbol{s}(\boldsymbol{z})\} + \boldsymbol{s}(\boldsymbol{z})\boldsymbol{s}(\boldsymbol{z})^{\mathsf{T}}$. Thus, the Hessian of the objective function in (2.21) is

$$\boldsymbol{H}(\boldsymbol{z}_m) = -\mathrm{diag}\{\boldsymbol{s}(\boldsymbol{z}_m)\} + \boldsymbol{s}(\boldsymbol{z}_m)\boldsymbol{s}(\boldsymbol{z}_m)^{\mathsf{T}} - [\boldsymbol{Q}^{\mathbf{p}}_m]^{-1}. \tag{2.30}$$

2.2.2 Multinomial logistic regression via MMSE-HyGAMP

In order to apply MMSE-HyGAMP to multinomial logistic regression, at each input node we must compute the posterior mean and covariance matrix of the approximate posterior distribution given in (2.3) for an appropriately chosen prior, $p_{\mathbf{x}}$.

Then, at each output node we must compute the posterior mean and covariance matrix of the approximate posterior distribution given in (2.4), where $p_{\mathsf{y}|\mathsf{z}}$ is the multinomial logistic activation function given in (2.5).

Input estimators: inference of $\widehat{\boldsymbol{x}}_n$

In MMSE-HyGAMP, the computation required at line 6 of Algorithm 1, given (2.3), is

$$\widehat{\boldsymbol{x}}_n = \mathrm{E}\{\mathsf{x}_n \,|\, \mathsf{r}_n = \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}^{\mathsf{r}}_n\} \tag{2.31}$$

$$= \frac{\int_{\mathbb{R}^D} \boldsymbol{x}_n p_{\mathsf{x}}(\boldsymbol{x}_n) \mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}^{\mathsf{r}}_n)\, \mathrm{d}\boldsymbol{x}_n}{\int_{\mathbb{R}^D} p_{\mathsf{x}}(\boldsymbol{x}_n) \mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}^{\mathsf{r}}_n)\, \mathrm{d}\boldsymbol{x}_n} \tag{2.32}$$

and the computation required at line 7 is

$$\boldsymbol{Q}^{\mathsf{x}}_n = \mathrm{var}\,\{\mathsf{x}_n \,|\, \mathsf{r}_n = \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}^{\mathsf{r}}_n\} \tag{2.33}$$

$$= \frac{\int_{\mathbb{R}^D} \boldsymbol{x}_n \boldsymbol{x}_n^{\mathsf{T}} p_{\mathsf{x}}(\boldsymbol{x}_n) \mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}^{\mathsf{r}}_n)\, \mathrm{d}\boldsymbol{x}_n}{\int_{\mathbb{R}^D} p_{\mathsf{x}}(\boldsymbol{x}_n) \mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}^{\mathsf{r}}_n)\, \mathrm{d}\boldsymbol{x}_n} - \widehat{\boldsymbol{x}}_n \widehat{\boldsymbol{x}}_n^{\mathsf{T}}, \tag{2.34}$$

with the iteration index t omitted here for simplicity.

We choose to use a multivariate Bernoulli-Gaussian distribution for p_{x}, given by

$$p_{\mathsf{x}}(\boldsymbol{x}_n) = \lambda_0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_0^x, \boldsymbol{\Sigma}_0^x) + (1 - \lambda_0)\delta(\boldsymbol{x}_n), \quad \lambda_0 \in (0, 1]. \tag{2.35}$$

Depending on λ_0, this prior distribution can encourage MMSE-HyGAMP to learn an approximately sparse $\widehat{\boldsymbol{X}}$.

Additionally, with this prior distribution, (2.32) and (2.34) have closed-form solutions. Given (2.35), the normalizing constant is

$$C_n \triangleq \int_{\mathbb{R}^D} p_{\mathbf{x}}(\boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})\,\mathrm{d}\boldsymbol{x}_n \tag{2.36}$$

$$= \int_{\mathbb{R}^D} \big(\lambda_0\mathcal{N}(\boldsymbol{x}_n;\boldsymbol{\mu}_0^x,\boldsymbol{\Sigma}_0^x) + (1-\lambda_0)\delta(\boldsymbol{x}_n)\big)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})\,\mathrm{d}\boldsymbol{x}_n \tag{2.37}$$

$$= \int_{\mathbb{R}^D} \lambda_0\mathcal{N}(\boldsymbol{x}_n;\boldsymbol{\mu}_0^x,\boldsymbol{\Sigma}_0^x)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})\,\mathrm{d}\boldsymbol{x}_n + \int_{\mathbb{R}^D} (1-\lambda_0)\delta(\boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})\,\mathrm{d}\boldsymbol{x}_n \tag{2.38}$$

$$= \lambda_0\mathcal{N}(\mathbf{0};\boldsymbol{\mu}_0^x-\widehat{\boldsymbol{r}}_n,\boldsymbol{\Sigma}_0^x+\boldsymbol{Q}_n^{\mathbf{r}}) + (1-\lambda_0)\mathcal{N}(\mathbf{0};\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}}). \tag{2.39}$$

The first term in (2.39) comes from (2.12), while the second term comes from the sifting property [25]. Then, given (2.32), (2.35), and (2.39),

$$\widehat{\boldsymbol{x}}_n = C_n^{-1}\int_{\mathbb{R}^D} \boldsymbol{x}_n p_{\mathbf{x}}(\boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})\,\mathrm{d}\boldsymbol{x}_n \tag{2.40}$$

$$= C_n^{-1}\int_{\mathbb{R}^D} \boldsymbol{x}_n\lambda_0\mathcal{N}(\boldsymbol{x}_n;\boldsymbol{\mu}_0^x,\boldsymbol{\Sigma}_0^x)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})\,\mathrm{d}\boldsymbol{x}_n + C_n^{-1}\int_{\mathbb{R}^D} \boldsymbol{x}_n(1-\lambda_0)\delta(\boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})\,\mathrm{d}\boldsymbol{x}_n \tag{2.41}$$

$$= C_n^{-1}\lambda_0\mathcal{N}(\mathbf{0};\boldsymbol{\mu}_0^x-\widehat{\boldsymbol{r}}_n,\boldsymbol{\Sigma}_0^x+\boldsymbol{Q}_n^{\mathbf{r}})([\boldsymbol{\Sigma}_0^x]^{-1}+[\boldsymbol{Q}_n^{\mathbf{r}}]^{-1})^{-1}([\boldsymbol{\Sigma}_0^x]^{-1}\boldsymbol{\mu}_0^x+[\boldsymbol{Q}_n^{\mathbf{r}}]^{-1}\widehat{\boldsymbol{r}}_n) \tag{2.42}$$

$$= \frac{([\boldsymbol{\Sigma}_0^x]^{-1}+[\boldsymbol{Q}_n^{\mathbf{r}}]^{-1})^{-1}([\boldsymbol{\Sigma}_0^x]^{-1}\boldsymbol{\mu}_0^x+[\boldsymbol{Q}_n^{\mathbf{r}}]^{-1}\widehat{\boldsymbol{r}}_n)}{1+\frac{1-\lambda_0}{\lambda_0}\frac{\mathcal{N}(\mathbf{0};\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})}{\mathcal{N}(\mathbf{0};\boldsymbol{\mu}_0^x-\widehat{\boldsymbol{r}}_n,\boldsymbol{\Sigma}_0^x+\boldsymbol{Q}_n^{\mathbf{r}})}}. \tag{2.43}$$

The first term in (2.41) evaluates to (2.42) via (2.12), while the second term in (2.41) evaluates to zero.

Lastly, given (2.34), (2.35), and (2.39),

$$\boldsymbol{Q}_n^{\mathbf{x}} = C_n^{-1}\int_{\mathbb{R}^D} \boldsymbol{x}_n\boldsymbol{x}_n^{\mathsf{T}} p_{\mathbf{x}}(\boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})\,\mathrm{d}\boldsymbol{x}_n - \widehat{\boldsymbol{x}}_n\widehat{\boldsymbol{x}}_n^{\mathsf{T}} \tag{2.44}$$

$$= C_n^{-1}\int_{\mathbb{R}^D} \boldsymbol{x}_n\boldsymbol{x}_n^{\mathsf{T}}\lambda_0\mathcal{N}(\boldsymbol{x}_n;\boldsymbol{\mu}_0^x,\boldsymbol{\Sigma}_0^x)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})\,\mathrm{d}\boldsymbol{x}_n + C_n^{-1}\int_{\mathbb{R}^D} \boldsymbol{x}_n\boldsymbol{x}_n^{\mathsf{T}}(1-\lambda_0)\delta(\boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}_n;\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathbf{r}})\,\mathrm{d}\boldsymbol{x}_n - \widehat{\boldsymbol{x}}_n\widehat{\boldsymbol{x}}_n^{\mathsf{T}}. \tag{2.45}$$

The first term in (2.45) evaluates to the scaled, second moment of a multivariate Gaussian distribution via (2.12). The scaling is given by $C_n^{-1}\mathcal{N}(\mathbf{0}; \boldsymbol{\mu}_0^x - \widehat{\boldsymbol{r}}_n, \boldsymbol{\Sigma}_0^x + \boldsymbol{Q}_n^{\mathsf{r}})$ and the second moment is given by $\boldsymbol{S} + \boldsymbol{m}\boldsymbol{m}^{\mathsf{T}}$ where $\boldsymbol{m} = ([\boldsymbol{\Sigma}_0^x]^{-1} + [\boldsymbol{Q}_n^{\mathsf{r}}]^{-1})^{-1}([\boldsymbol{\Sigma}_0^x]^{-1}\boldsymbol{\mu}_0^x + [\boldsymbol{Q}_n^{\mathsf{r}}]^{-1}\widehat{\boldsymbol{r}}_n)$ and $\boldsymbol{S} = ([\boldsymbol{\Sigma}_0^x]^{-1} + [\boldsymbol{Q}_n^{\mathsf{r}}]^{-1})^{-1}$. The middle term in (2.45) evaluates to zero. This results in

$$\boldsymbol{Q}_n^{\mathsf{x}} = \frac{\boldsymbol{S} + \boldsymbol{m}\boldsymbol{m}^{\mathsf{T}}}{1 + \frac{1-\lambda_0}{\lambda_0}\frac{\mathcal{N}(\mathbf{0};\widehat{\boldsymbol{r}}_n,\boldsymbol{Q}_n^{\mathsf{r}})}{\mathcal{N}(\mathbf{0};\boldsymbol{\mu}_0^x-\widehat{\boldsymbol{r}}_n,\boldsymbol{\Sigma}_0^x+\boldsymbol{Q}_n^{\mathsf{r}})}} - \widehat{\boldsymbol{x}}_n\widehat{\boldsymbol{x}}_n^{\mathsf{T}}. \tag{2.46}$$

Output estimators: inference of $\widehat{\boldsymbol{z}}_m$

In MMSE-HyGAMP, the computation required at line 15 of Algorithm 1, given (2.4), is

$$\widehat{\boldsymbol{z}}_m = \mathrm{E}\{\mathsf{z}_m \,|\, \mathsf{y}_m = y_m, \mathsf{p}_m = \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}_m^{\mathsf{p}}\} \tag{2.47}$$

$$= \frac{\int_{\mathbb{R}^D} \boldsymbol{z}_m p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}_m^{\mathsf{p}})\,\mathrm{d}\boldsymbol{z}_m}{\int_{\mathbb{R}^D} p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}_m^{\mathsf{p}})\,\mathrm{d}\boldsymbol{z}_m} \tag{2.48}$$

and the computation required at line 16 is

$$\boldsymbol{Q}_m^{\mathsf{z}} = \mathrm{var}\{\mathsf{z}_m \,|\, \mathsf{y}_m = y_m, \mathsf{p}_m = \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}_m^{\mathsf{p}}\} \tag{2.49}$$

$$= \frac{\int_{\mathbb{R}^D} \boldsymbol{z}_m\boldsymbol{z}_m^{\mathsf{T}} p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}_m^{\mathsf{p}})\,\mathrm{d}\boldsymbol{z}_m}{\int_{\mathbb{R}^D} p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}_m^{\mathsf{p}})\,\mathrm{d}\boldsymbol{z}_m} - \widehat{\boldsymbol{z}}_m\widehat{\boldsymbol{z}}_m^{\mathsf{T}}, \tag{2.50}$$

with the iteration index t again omitted here for simplicity.

The problems in (2.47) and (2.49) are not solvable in closed-form, but they can be approximated using, e.g., numerical integration or importance sampling [1, §11.1.4]. In numerical integration, we use a D-dimensional hyper-rectangular grid where each dimension d has J grid points uniformly spaced between $\left[\widehat{p}_{md} - \alpha\sqrt{[\boldsymbol{Q}_m^{\mathsf{p}}]_{dd}}, \widehat{p}_{md} + \alpha\sqrt{[\boldsymbol{Q}_m^{\mathsf{p}}]_{dd}}\right]$.

If we use importance sampling, we draw J independent samples $\{\widetilde{\boldsymbol{z}}_m[j]\}_{j=1}^J$ from $\mathcal{N}(\boldsymbol{z}_m; \widehat{\boldsymbol{p}}_m, \boldsymbol{Q}_m^{\mathsf{p}})$. Then, the approximations become

$$\widehat{\boldsymbol{z}}_m \approx \frac{\sum_{j=1}^J \widetilde{\boldsymbol{z}}_m[j] p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \widetilde{\boldsymbol{z}}_m[j])}{\sum_{j=1}^J p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \widetilde{\boldsymbol{z}}_m[j])} \tag{2.51}$$

and

$$\boldsymbol{Q}_m^{\mathsf{z}} \approx \frac{\sum_{j=1}^J \widetilde{\boldsymbol{z}}_m[j] \widetilde{\boldsymbol{z}}_m[j]^{\mathsf{T}} p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \widetilde{\boldsymbol{z}}_m[j])}{\sum_{j=1}^J p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \widetilde{\boldsymbol{z}}_m[j])} - \widehat{\boldsymbol{z}}_m \widehat{\boldsymbol{z}}_m^{\mathsf{T}}. \tag{2.52}$$

2.3 Conclusion

In this chapter, we applied Hybrid-GAMP to the MAP and MMSE multinomial logistic regression problems in (1.12) and (1.14). Then we showed how to compute the non-trivial steps in the algorithm, some of which can be solved in closed-form, some of which rely on approximations, and some of which can be solved iteratively using an optimization algorithm.

Chapter 3: Simplified Hybrid-GAMP for Multinomial Logistic Regression

In Chapter 2 we applied HyGAMP to a MAP and an MMSE formulation of the multinomial logistic regression problem. However, a direct application of HyGAMP to multinomial logistic regression is computationally costly due in part to the many matrix inverses and also the input and output estimators.

Thus, in this Chapter, we propose a *simplified* Hybrid-GAMP (SHyGAMP) algorithm for sparse multinomial logistic regression, whose complexity is greatly reduced. The simplification itself is rather straightforward: we constrain the covariance matrices $\boldsymbol{Q}_n^{\mathsf{r}}$, $\boldsymbol{Q}_n^{\mathsf{x}}$, $\boldsymbol{Q}_m^{\mathsf{p}}$, and $\boldsymbol{Q}_m^{\mathsf{z}}$ to be diagonal. In other words,

$$\boldsymbol{Q}_n^{\mathsf{r}} = \operatorname{diag}\left\{q_{n1}^{\mathsf{r}}, \ldots, q_{nD}^{\mathsf{r}}\right\}, \tag{3.1}$$

and similar for $\boldsymbol{Q}_n^{\mathsf{x}}$, $\boldsymbol{Q}_m^{\mathsf{p}}$, and $\boldsymbol{Q}_m^{\mathsf{z}}$. 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 computationally simpler methods than those described in Chapter 2, as we detail in the squeal.

3.1 Scalar variance SHyGAMP

We further approximate the SHyGAMP algorithm using the *scalar variance* approximation from [11], which reduces the memory and complexity of the algorithm. The scalar variance approximation first approximates $\{q^{\mathsf{x}}_{nd}\}$ by a version invariant to both n and d, i.e.,

$$q^{\mathsf{x}} \triangleq \frac{1}{ND}\sum_{n=1}^{N}\sum_{d=1}^{D} q^{\mathsf{x}}_{nd}. \tag{3.2}$$

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

$$q^{\mathsf{p}}_{md} \approx \sum_{n=1}^{N} A^2_{mn} q^{\mathsf{x}} \tag{3.3}$$

$$\approx \frac{\|\boldsymbol{A}\|_F^2}{M} q^{\mathsf{x}} \tag{3.4}$$

$$\triangleq q^{\mathsf{p}}, \tag{3.5}$$

where $||\cdot||_F$ is the Frobenius norm. The approximation in (3.4), after precomputing $\|\boldsymbol{A}\|_F^2$, reduces the complexity of line 9 from $O(ND)$ to $O(1)$. We next assume

$$q^{\mathsf{s}} \triangleq \frac{1}{MD}\sum_{m=1}^{M}\sum_{d=1}^{D} q^{\mathsf{s}}_{md}, \tag{3.6}$$

and in line 20 we use the approximation

$$q^{\mathsf{r}}_{nd} \approx \left[\sum_{m=1}^{M} A^2_{mn} q^{\mathsf{s}}\right]^{-1} \tag{3.7}$$

$$\approx \frac{N}{q^{\mathsf{s}}\|\boldsymbol{A}\|_F^2} \tag{3.8}$$

$$\triangleq q^{\mathsf{r}}. \tag{3.9}$$

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

3.2 Multinomial logistic regression via MAP-SHyGAMP

In order to perform multinomial logistic regression with MAP-SHyGAMP we follow the same procedure as MAP-HyGAMP, detailed in Section 2.2.1, but with the constraint that all $\boldsymbol{Q}^{\mathsf{r}}$, $\boldsymbol{Q}^{\mathsf{x}}$, $\boldsymbol{Q}^{\mathsf{p}}$, and $\boldsymbol{Q}^{\mathsf{z}}$ are diagonal.

3.2.1 Input estimators: inference of $\widehat{\boldsymbol{x}}_n$

Recall from Section 2.2.1, we must compute

$$\widehat{\boldsymbol{x}}_n = \underset{\boldsymbol{x}_n \in \mathbb{R}^D}{\arg\max} \left\{ \log p_{\mathsf{x}|\mathsf{r}}(\boldsymbol{x}_n \,|\, \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}^{\mathsf{r}}_n) \right\} \tag{3.10}$$

$$= \underset{\boldsymbol{x}_n \in \mathbb{R}^D}{\arg\max} \left\{ \log p_{\mathsf{x}}(\boldsymbol{x}_n) - \frac{1}{2}(\boldsymbol{x}_n - \widehat{\boldsymbol{r}}_n)^{\mathsf{T}}[\boldsymbol{Q}^{\mathsf{r}}_n]^{-1}(\boldsymbol{x}_n - \widehat{\boldsymbol{r}}_n) \right\} \tag{3.11}$$

and

$$\boldsymbol{Q}^{\mathsf{x}}_n = \left[-\frac{\partial^2}{\partial \boldsymbol{x}_n^2} \log p_{\mathsf{x}\,|\,\mathsf{r}}(\widehat{\boldsymbol{x}}_n \,|\, \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}^{\mathsf{r}}_n) \right]^{-1}. \tag{3.12}$$

Under the assumption of diagonal $\boldsymbol{Q}^{\mathsf{r}}_n$ and a separable prior, i.e., $p_{\boldsymbol{x}}(\boldsymbol{x}_n) = \prod_{d=1}^{D} p_{\mathsf{x}}(x_{nd})$, the problems in (3.11) and (3.12) decouple into D scalar problems, shown in (3.14) and (3.15).

$$\widehat{x}_{nd} = \underset{x}{\arg\max} \left\{ \log p_{\mathsf{x}\,|\,\mathsf{r}}(x_{nd} \,|\, \widehat{r}_{nd}; q^{\mathsf{r}}_{nd}) \right\} \tag{3.13}$$

$$= \underset{x}{\arg\max} \left\{ \log p_{\mathsf{x}}(x_{nd}) - \frac{1}{2q^{\mathsf{r}}_{nd}}(x_{nd} - \widehat{r}_{nd})^2 \right\} \tag{3.14}$$

and

$$q^{\mathsf{x}}_{nd} = \left[-\frac{d^2}{dx^2} \log p_{\mathsf{x}\,|\,\mathsf{r}}(x_{nd} \,|\, \widehat{r}_{nd}; q^{\mathsf{r}}_{nd}) \right]^{-1}. \tag{3.15}$$

Choice of prior In MAP-HyGAMP we used a vector-valued Gaussian or Laplacian prior on x_n. In MAP-SHyGAMP we use a scalar-valued Gaussian or Laplacian prior on x_{nd}, which correspond to the ℓ_2 norm or sparsity-promoting ℓ_1 norm constraint, respectively, on x_{nd}.

Gaussian Under a Gaussian prior, defined by $p_{\mathsf{x}}(x_{nd}) = \mathcal{N}(x_{nd}; \mu_0, \sigma_0^2)$, the solution to (3.14) is the scalar equivalent to (2.11), given by

$$\widehat{x}_{nd} = ([\sigma_0^2]^{-1} + [q_{nd}^{\mathsf{r}}]^{-1})^{-1}([\sigma_0^2]^{-1}\mu_0 + [q_{nd}^{\mathsf{r}}]^{-1}\widehat{r}_{nd}). \tag{3.16}$$

The solution to (3.15) is

$$q_{nd}^{\mathsf{x}} = ([\sigma_0^2]^{-1} + [q_{nd}^{\mathsf{r}}]^{-1})^{-1}. \tag{3.17}$$

Laplacian Under a Laplacian prior, defined by

$$p_{\mathsf{x}}(x_{nd}) \propto \exp(-\lambda|x_{nd}|), \tag{3.18}$$

the solution to (3.14) is shown in (3.19), which is known as soft-thresholding.

$$\widehat{x}_{nd} = \operatorname{sgn}(\widehat{r}_{nd}) \max\{0, |\widehat{r}_{nd}| - \lambda q_{nd}^{\mathsf{r}}\}, \tag{3.19}$$

where $\operatorname{sgn}(\cdot)$ is the sign function. Contrast (3.19) to the solution to (2.9) under a vector-valued Laplacian prior, given in (2.19). In the scalar case, the exact solution is given in a single step, while in the vector case only an approximate solution is found through an MM procedure, which may require many iterations (each of which requires a $D \times D$ matrix inverse).

Since the second derivative of $|x|$ is zero everywhere except the $x = 0$, for which it is undefined, $q_{nd}^{\mathsf{x}} = q_{nd}^{\mathsf{r}}$ except when $x_{nd} = 0$, in which case we set $q_{nd}^{\mathsf{x}} = 0$.

3.2.2 Output estimators: inference of $\widehat{\boldsymbol{z}}_m$

As shown in Section 2.2.1, we must compute

$$\widehat{\boldsymbol{z}}_m = \arg\max_{\boldsymbol{z}_m \in \mathbb{R}^D} \left\{ \log p_{\mathsf{z}\,|\,\mathsf{y},\mathsf{p}}(\boldsymbol{z}_m \,|\, y_m, \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}_m^{\mathsf{p}}) \right\} \tag{3.20}$$

$$= \arg\max_{\boldsymbol{z}_m \in \mathbb{R}^D} \left\{ \log p_{\mathsf{y}\,|\,\mathsf{z}}(y_m \,|\, \boldsymbol{z}_m) - \frac{1}{2}(\boldsymbol{z}_m - \widehat{\boldsymbol{p}}_m)^{\mathsf{T}}[\boldsymbol{Q}_m^{\mathsf{p}}]^{-1}(\boldsymbol{z}_m - \widehat{\boldsymbol{p}}_m) \right\} \tag{3.21}$$

and

$$\boldsymbol{Q}_m^{\mathsf{z}} = \left[-\frac{\partial^2}{\partial \boldsymbol{z}_m^2} \left\{ \log p_{\mathsf{z}\,|\,\mathsf{y},\mathsf{p}}(\widehat{\boldsymbol{z}}_m \,|\, y_m, \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}_m^{\mathsf{p}}) \right\} \right]^{-1}. \tag{3.22}$$

Unlike the simplifications resulting from diagonal $\boldsymbol{Q}_n^{\mathsf{r}}$ in Section 2.2.1, assuming diagonal $\boldsymbol{Q}_m^{\mathsf{p}}$ does not allow (3.21) to decouple into D scalar problems due to the form of $p_{\mathsf{y}\,|\,\mathsf{z}}$. In Section 2.2.1, we used Newton's method to solve (3.21). In MAP-SHyGAMP we choose to use component-wise Newton's method. We note that we could have used component-wise Newton's method in MAP-HyGAMP, but the output estimator in MAP-HyGAMP is not a bottleneck compared to the linear steps and the input estimators.

3.2.3 Parameter selection for MAP-SHyGAMP

The Laplacian prior in (3.18) requires the specification (i.e., tuning) of the scale parameter λ, which controls the sparsity of the estimated weight vector. Traditionally, λ is tuned via cross-validation, which involves constructing a set of hypotheses for λ and then training a classifier for each one. We avoid this computationally intensive procedure by leveraging the SURE-AMP framework from [26]. SURE-AMP adjusts λ to minimize the Stein's unbiased risk estimate (SURE) of the weight-vector MSE.

First, we give a brief description of SURE. Suppose that the signal of interest, $\{x_{nd}\}$, is a realization of i.i.d. random variables $\{\mathsf{x}_{nd}\}$, and that to estimate this signal we are given access to AWGN corrupted observations

$$\widehat{r}_{nd} = x_{nd} + \sqrt{q^{\mathsf{r}}} w_{nd}, \tag{3.23}$$

with known $q^{\mathsf{r}} > 0$ and unit-variance AWGN $\{w_{nd}\}$. For an arbitrary scalar estimation procedure $\widehat{x}_{nd} = f(\widehat{r}_{nd}, q^{\mathsf{r}}; \lambda)$, the SURE estimate of the MSE(λ) $\triangleq \sum_{n,d} \mathrm{E}\{[\widehat{\mathsf{x}}_{nd} -$

$\mathsf{x}_{nd}]^2\}$ is defined as

$$\mathrm{SURE}(\widehat{\boldsymbol{R}};\lambda) \triangleq \sum_{n,d}\Big(g^2\big(\widehat{r}_{nd},q^{\mathsf{r}};\lambda\big)+2q^{\mathsf{r}}g'(\widehat{r}_{nd},q^{\mathsf{r}};\lambda)+q^{\mathsf{r}}\Big), \tag{3.24}$$

for $g(\widehat{r}_{nd},q^{\mathsf{r}};\lambda) \triangleq f(\widehat{r}_{nd},q^{\mathsf{r}};\lambda)-\widehat{r}_{nd}$. The key property of SURE is that

$$\mathrm{E}\left\{\mathrm{SURE}(\widehat{\mathbf{R}};\lambda)\right\} = \mathrm{MSE}(\lambda), \tag{3.25}$$

i.e., it is an unbiased estimate of the MSE that can be calculated with an arbitrary estimation function $f(\cdot)$ and without any knowledge of the statistics of x_{nd}.

In [26], it was noticed that the SURE assumptions are perfectly satisfied by the AMP denoiser inputs, and thus it was proposed to set λ at the value minimizing $\mathrm{SURE}(\widehat{\boldsymbol{R}};\lambda)$, and thereby approximately minimize MSE. Conveniently, the SURE assumptions also match the scalar-variance SHyGAMP model from Section 3.1, and thus we propose to do the same. In particular, we propose to set

$$\widehat{\lambda} = \arg\min_{\lambda}\sum_{n=1}^{N}\sum_{d=1}^{D} g^2\big(\widehat{r}_{nd},q^{\mathsf{r}};\lambda\big)+2q^{\mathsf{r}}g'(\widehat{r}_{nd},q^{\mathsf{r}};\lambda). \tag{3.26}$$

Recall that (3.19) specifies $f(\cdot)$, after which it follows that

$$g^2(\widehat{r}_{nd},q^{\mathsf{r}};\lambda) = \begin{cases}\lambda^2(q^{\mathsf{r}})^2 & \text{if } |\widehat{r}_{nd}|>\lambda q^{\mathsf{r}}\\ \widehat{r}_{nd}^2 & \text{otherwise}\end{cases} \tag{3.27}$$

$$g'(\widehat{r}_{nd},q^{\mathsf{r}};\lambda) = \begin{cases}-1 & \text{if } |\widehat{r}_{nd}|<\lambda q^{\mathsf{r}}\\ 0 & \text{otherwise}\end{cases}. \tag{3.28}$$

Solving (3.26) for this $g(\cdot)$ is non-trivial because the cost in (3.26) 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.

We propose a novel approach to SURE minimization where the empirical average in (3.26) is replaced by a statistical average, i.e.,

$$\widehat{\lambda} = \arg\min_{\lambda}\mathrm{E}\left\{g^2\big(\widehat{\mathsf{r}},q^{\mathsf{r}};\lambda\big)+2q^{\mathsf{r}}g'(\widehat{\mathsf{r}},q^{\mathsf{r}};\lambda)\right\}, \tag{3.29}$$

where the random variable $\widehat{\mathsf{r}}$ is modeled as a Gaussian-Mixture (GM) whose parameters are fitted to $\{\widehat{r}_{nd}\}$. As a result, the cost function in (3.29) is smooth and (we conjecture) unimodal. In practice, we use the standard EM approach to GM fitting and we find that relatively few (e.g., three) mixture terms suffice.

For the GM model $p_{\mathsf{r}}(r) = \sum_{l=1}^{L} \alpha_l \mathcal{N}(r; \theta_l, \sigma_l^2)$, the cost in (3.29) reduces to

$$\mathrm{E}\{g^2(\mathsf{r}, q^{\mathsf{r}}; \lambda) + 2q^{\mathsf{r}} g'(\mathsf{r}, q^{\mathsf{r}}; \lambda)\} = \int p_{\mathsf{r}}(r)\big(g^2(r, q^{\mathsf{r}}; \lambda) + 2q^{\mathsf{r}} g'(r, q^{\mathsf{r}}; \lambda)\big)\,\mathrm{d}r \tag{3.30}$$

$$= \int_{-\infty}^{-\lambda q^{\mathsf{r}}} p_{\mathsf{r}}(r)(\lambda^2 (q^{\mathsf{r}})^2)\,\mathrm{d}r + \int_{-\lambda q^{\mathsf{r}}}^{\lambda q^{\mathsf{r}}} p_{\mathsf{r}}(r)(r^2 - 2q^{\mathsf{r}})\,\mathrm{d}r + \int_{\lambda q^{\mathsf{r}}}^{\infty} p_{\mathsf{r}}(r)(\lambda^2 (q^{\mathsf{r}})^2)\,\mathrm{d}r \tag{3.31}$$

$$= \Pr\{r < -\lambda q^{\mathsf{r}}\}\lambda^2 (q^{\mathsf{r}})^2 + \int_{-\lambda q^{\mathsf{r}}}^{\lambda q^{\mathsf{r}}} r^2 p_{\mathsf{r}}(r)\,\mathrm{d}r - 2q^{\mathsf{r}} \Pr\{|r| < \lambda q^{\mathsf{r}}\} + \Pr\{r > \lambda q^{\mathsf{r}}\}\lambda^2 (q^{\mathsf{r}})^2 \tag{3.32}$$

$$\begin{aligned} &= \lambda^2 (q^{\mathsf{r}})^2 \sum_{l=1}^{L} \alpha_l \Phi\left(\frac{-\lambda q^{\mathsf{r}} - \theta_l}{\sigma_l}\right) + \int_{-\lambda q^{\mathsf{r}}}^{\lambda q^{\mathsf{r}}} r^2 p_{\mathsf{r}}(r)\,\mathrm{d}r \\ &\quad - 2q^{\mathsf{r}} \left(\sum_{l=1}^{L} \alpha_l \Phi\left(\frac{\lambda q^{\mathsf{r}} - \theta_l}{\sigma_l}\right) - \sum_{l=1}^{L} \alpha_l \Phi\left(\frac{-\lambda q^{\mathsf{r}} - \theta_l}{\sigma_l}\right) \right) \\ &\quad + \lambda^2 (q^{\mathsf{r}})^2 \left(1 - \sum_{l=1}^{L} \alpha_l \Phi\left(\frac{\lambda q^{\mathsf{r}} - \theta_l}{\sigma_l}\right) \right), \end{aligned} \tag{3.33}$$

where

$$\int_{-\lambda q^{\mathsf{r}}}^{\lambda q^{\mathsf{r}}} r^2 p_{\mathsf{r}}(r)\,\mathrm{d}r = \sum_{l=1}^{L} \alpha_l \int_{-\lambda q^{\mathsf{r}}}^{\lambda q^{\mathsf{r}}} r^2 \mathcal{N}(r; \theta_l, \sigma_l^2)\,\mathrm{d}r \tag{3.34}$$

$$\begin{aligned} &= \sum_{l=1}^{L} \alpha_l \Big(\Phi(b_l) - \Phi(a_l)\Big) \left[\sigma_l^2 \left(1 + \frac{a_l \phi(a_l) - b_l \phi(b_l)}{\Phi(b_l) - \Phi(a_l)} - \left(\frac{\phi(a_l) - \phi(b_l)}{\Phi(b_l) - \Phi(a_l)} \right)^2 \right) + \right. \\ &\quad \left. \left(\theta_l + \sigma_l \frac{\phi(a_l) - \phi(b_l)}{\Phi(b_l) - \Phi(a_l)} \right)^2 \right], \end{aligned} \tag{3.35}$$

such that $a_l \triangleq (-\lambda q^{\mathsf{r}} - \theta_l)/\sigma_l$, and $b_l \triangleq (\lambda q^{\mathsf{r}} - \theta_l)/\sigma_l$. Note that (3.35) can be derived by applying the second moment of a truncated normal distribution to (3.34), which requires the introduction of the scale term of $\big(\Phi(b_l) - \Phi(a_l)\big)$ in the sum to make it a valid pdf.

We then propose to solve (3.29) using the bisection method. The use of bisection follows from our conjecture that the cost in (3.29) is unimodal, implying that its derivative w.r.t. λ has a unique root. Thus, we use bisection to find the root of $\frac{\mathrm{d}}{\mathrm{d}\lambda}\,\mathrm{E}\{g^2(\mathsf{r},q^{\mathsf{r}};\lambda)+2cg'(\mathsf{r},q^{\mathsf{r}};\lambda)\}$. To find an expression for this derivative, we first rewrite (3.31) as

$$\begin{aligned}
&\mathrm{E}\{g^2(\mathsf{r},q^{\mathsf{r}};\lambda)+2q^{\mathsf{r}}g'(\mathsf{r},q^{\mathsf{r}};\lambda)\}\\
&=\int_{-\infty}^{-\lambda q^{\mathsf{r}}} p_{\mathsf{r}}(r)(\lambda^2(q^{\mathsf{r}})^2)\,\mathrm{d}r+\left(\int_{-\infty}^{\lambda q^{\mathsf{r}}} p_{\mathsf{r}}(r)(r^2-2q^{\mathsf{r}})\,\mathrm{d}r-\int_{-\infty}^{-\lambda q^{\mathsf{r}}} p_{\mathsf{r}}(r)(r^2-2q^{\mathsf{r}})\,\mathrm{d}r\right)\\
&\quad+\int_{-\infty}^{\infty} p_{\mathsf{r}}(r)\lambda^2(q^{\mathsf{r}})^2\,\mathrm{d}r-\int_{-\infty}^{\lambda q^{\mathsf{r}}} p_{\mathsf{r}}(r)\lambda^2(q^{\mathsf{r}})^2\,\mathrm{d}r, \qquad (3.36)
\end{aligned}$$

from which it follows that

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\lambda}\,\mathrm{E}\{g^2(\mathsf{r},q^{\mathsf{r}};\lambda)+2q^{\mathsf{r}}g'(\mathsf{r},q^{\mathsf{r}};\lambda)\}\\
&=-\lambda^2(q^{\mathsf{r}})^3p_{\mathsf{r}}(-\lambda q^{\mathsf{r}})+2\lambda(q^{\mathsf{r}})^2\Pr\{\mathsf{r}<-\lambda q^{\mathsf{r}}\}\\
&\quad+q^{\mathsf{r}}p_{\mathsf{r}}(\lambda q^{\mathsf{r}})(\lambda^2(q^{\mathsf{r}})^2-2q^{\mathsf{r}})+q^{\mathsf{r}}p_{\mathsf{r}}(-\lambda q^{\mathsf{r}})(\lambda^2(q^{\mathsf{r}})^2-2q^{\mathsf{r}})\\
&\quad+2\lambda(q^{\mathsf{r}})^2-\lambda^2(q^{\mathsf{r}})^3p_{\mathsf{r}}(\lambda q^{\mathsf{r}})-2\lambda(q^{\mathsf{r}})^2\Pr\{\mathsf{r}<\lambda q^{\mathsf{r}}\} \qquad (3.37)\\
&=2\lambda(q^{\mathsf{r}})^2\big[1-\Pr\{-\lambda q^{\mathsf{r}}<\mathsf{r}<\lambda q^{\mathsf{r}}\}\big]-\big[p_{\mathsf{r}}(\lambda q^{\mathsf{r}})+p_{\mathsf{r}}(-\lambda q^{\mathsf{r}})\big]2(q^{\mathsf{r}})^2 \qquad (3.38)\\
&=2(q^{\mathsf{r}})^{\frac{3}{2}}\left[\lambda\sqrt{q^{\mathsf{r}}}\left(1-\Pr\left\{-\lambda\sqrt{q^{\mathsf{r}}}<\frac{\mathsf{r}}{\sqrt{q^{\mathsf{r}}}}<\lambda\sqrt{q^{\mathsf{r}}}\right\}\right)-\left(p_{\mathsf{r}/\sqrt{q^{\mathsf{r}}}}(\lambda\sqrt{q^{\mathsf{r}}})+p_{\mathsf{r}/\sqrt{q^{\mathsf{r}}}}(-\lambda\sqrt{q^{\mathsf{r}}})\right)\right]. \qquad (3.39)
\end{aligned}$$

Note that we compute this estimation of λ at each iteration of Algorithm 1, immediately before line 3, i.e., prior to applying the input estimator.

3.3 Multinomial logistic regression via MMSE-SHyGAMP

In order to perform multinomial logistic regression with MMSE-SHyGAMP, we follow a similar procedure as with MMSE-HyGAMP, as detailed in Section 2.2.2.

Recall that, in SHyGAMP, we assume $\boldsymbol{Q}_n^{\mathsf{r}}$ and $\boldsymbol{Q}_m^{\mathsf{p}}$ are diagonal, and we refrain from computing the off-diagonal terms in $\boldsymbol{Q}_n^{\mathsf{x}}$ and $\boldsymbol{Q}_m^{\mathsf{z}}$.

3.3.1 Input estimators: inference of $\widehat{\boldsymbol{x}}_n$

As shown in Section 2.2.2, we must compute

$$\widehat{\boldsymbol{x}}_n = \mathrm{E}\{\mathsf{x}_n \,|\, \mathsf{r}_n = \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}_n^{\mathsf{r}}\} \tag{3.40}$$

$$= \frac{\int_{\mathbb{R}^D} \boldsymbol{x}_n p_{\mathsf{x}}(\boldsymbol{x}_n) \mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}_n^{\mathsf{r}}) \,\mathrm{d}\boldsymbol{x}_n}{\int_{\mathbb{R}^D} p_{\mathsf{x}}(\boldsymbol{x}_n) \mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}_n^{\mathsf{r}}) \,\mathrm{d}\boldsymbol{x}_n} \tag{3.41}$$

and

$$\boldsymbol{Q}_n^{\mathsf{x}} = \mathrm{var}\,\{\mathsf{x}_n \,|\, \mathsf{r}_n = \widehat{\boldsymbol{r}}_n; \boldsymbol{Q}_n^{\mathsf{r}}\} \tag{3.42}$$

$$= \frac{\int_{\mathbb{R}^D} \boldsymbol{x}_n \boldsymbol{x}_n^{\mathsf{T}} p_{\mathsf{x}}(\boldsymbol{x}_n) \mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}_n^{\mathsf{r}}) \,\mathrm{d}\boldsymbol{x}_n}{\int_{\mathbb{R}^D} p_{\mathsf{x}}(\boldsymbol{x}_n) \mathcal{N}(\boldsymbol{x}_n; \widehat{\boldsymbol{r}}_n, \boldsymbol{Q}_n^{\mathsf{r}}) \,\mathrm{d}\boldsymbol{x}_n} - \widehat{\boldsymbol{x}}_n \widehat{\boldsymbol{x}}_n^{\mathsf{T}}. \tag{3.43}$$

Under the assumption of diagonal $\boldsymbol{Q}_n^{\mathsf{r}}$ and a separable prior, i.e., $p_{\boldsymbol{x}}(\boldsymbol{x}_n) = \prod_{d=1}^{D} p_{\mathsf{x}}(x_{nd})$, the problems in (3.41) and (3.43) decouple into D scalar problems, shown in (3.45) and (3.47):

$$\widehat{x}_{nd} = \mathrm{E}\{\mathsf{x}_{nd} \,|\, \mathsf{r}_{nd} = \widehat{r}_{nd}\} \tag{3.44}$$

$$= \frac{\int x_{nd}\, p_{\mathsf{x}}(x_{nd}) \mathcal{N}(x_{nd}; \widehat{r}_{nd}, q_{nd}^{\mathsf{r}}) \,\mathrm{d}x_{nd}}{\int p_{\mathsf{x}}(x_{nd}) \mathcal{N}(x_{nd}; \widehat{r}_{nd}, q_{nd}^{\mathsf{r}}) \,\mathrm{d}x_{nd}} \tag{3.45}$$

and

$$q_{nd}^{\mathsf{x}} = \mathrm{var}\{\mathsf{x}_{nd} \,|\, \mathsf{r}_{nd} = \widehat{r}_{nd}\} \tag{3.46}$$

$$= \frac{\int x_{nd}^2\, p_{\mathsf{x}}(x_{nd}) \mathcal{N}(x_{nd}; \widehat{r}_{nd}, q_{nd}^{\mathsf{r}}) \,\mathrm{d}x_{nd}}{\int p_{\mathsf{x}}(x_{nd}) \mathcal{N}(x_{nd}; \widehat{r}_{nd}, q_{nd}^{\mathsf{r}}) \,\mathrm{d}x_{nd}} - \widehat{x}_{nd}^2. \tag{3.47}$$

We choose p_{x} to be a scalar-valued Bernoulli-Gaussian distribution given by

$$p_{\mathsf{x}}(x_{nd}) = \lambda_0 \mathcal{N}(x_{nd}; \mu_0, \sigma_0^2) + (1 - \lambda_0)\delta(x_{nd}), \tag{3.48}$$

which yields the following closed-form solutions for $\widehat{x}_{nd}$ and q^{x}_{nd}:

$$\widehat{x}_{nd} = \frac{([\sigma_0^2]^{-1} + [q^{\mathsf{r}}_{nd}]^{-1})^{-1}([\sigma_0^2]^{-1}\mu_0 + [q^{\mathsf{r}}_{nd}]^{-1}\widehat{r}_{nd})}{1 + \frac{1-\lambda_0}{\lambda_0}\frac{\mathcal{N}(0;\widehat{r}_{nd},q^{\mathsf{r}}_{nd})}{\mathcal{N}(0;\mu_0-\widehat{r}_{nd},\sigma_0^2+q^{\mathsf{r}}_{nd})}} \tag{3.49}$$

and

$$q^{\mathsf{x}}_{nd} = \frac{\left(([\sigma_0^2]^{-1} + [q^{\mathsf{r}}_{nd}]^{-1})^{-1}([\sigma_0^2]^{-1}\mu_0 + [q^{\mathsf{r}}_{nd}]^{-1}\widehat{r}_{nd})\right)^2 + ([\sigma_0^2]^{-1} + [q^{\mathsf{r}}_{nd}]^{-1})^{-1}}{1 + \frac{1-\lambda_0}{\lambda_0}\frac{\mathcal{N}(0;\widehat{r}_{nd},q^{\mathsf{r}}_{nd})}{\mathcal{N}(0;\mu_0-\widehat{r}_{nd},\sigma_0^2+q^{\mathsf{r}}_{nd})}} - \widehat{x}^2_{nd}. \tag{3.50}$$

The derivations for (3.49) and (3.50) are simply the scalar version of the derivations for (2.43) and (2.45).

Note that (3.48) is not the same distribution as the multivariate Bernoulli-Gaussian given in (2.35), but this distribution allows the D-dimensional input estimation problem to be broken up into D scalar estimation problems, with the trade-off of not being able to enforce row-sparsity in $\boldsymbol{X}$.

3.3.2 Output estimators: inference of $\widehat{\boldsymbol{z}}_m$

As shown in Section 2.2.2, we must compute

$$\widehat{\boldsymbol{z}}_m = \mathrm{E}\{\mathbf{z}_m \,|\, \mathsf{y}_m = y_m, \mathbf{p}_m = \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}^{\mathsf{p}}_m\} \tag{3.51}$$

$$= \frac{\int_{\mathbb{R}^D} \boldsymbol{z}_m p_{\mathsf{y}\,|\,\mathbf{z}}(y_m \,|\, \boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}_m)\,\mathrm{d}\boldsymbol{z}_m}{\int_{\mathbb{R}^D} p_{\mathsf{y}\,|\,\mathbf{z}}(y_m \,|\, \boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}_m)\,\mathrm{d}\boldsymbol{z}_m} \tag{3.52}$$

and

$$\boldsymbol{Q}^{\mathsf{z}}_m = \mathrm{var}\{\mathbf{z}_m \,|\, \mathsf{y}_m = y_m, \mathbf{p}_m = \widehat{\boldsymbol{p}}_m; \boldsymbol{Q}^{\mathsf{p}}_m\} \tag{3.53}$$

$$= \frac{\int_{\mathbb{R}^D} \boldsymbol{z}_m\boldsymbol{z}_m^{\mathsf{T}} p_{\mathsf{y}\,|\,\mathbf{z}}(y_m \,|\, \boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}_m)\,\mathrm{d}\boldsymbol{z}_m}{\int_{\mathbb{R}^D} p_{\mathsf{y}\,|\,\mathbf{z}}(y_m \,|\, \boldsymbol{z}_m)\mathcal{N}(\boldsymbol{z}_m;\widehat{\boldsymbol{p}}_m,\boldsymbol{Q}^{\mathsf{p}}_m)\,\mathrm{d}\boldsymbol{z}_m} - \widehat{\boldsymbol{z}}_m\widehat{\boldsymbol{z}}_m^{\mathsf{T}}. \tag{3.54}$$

Unfortunately, just as in the MAP case, assuming diagonal $\boldsymbol{Q}^{\mathsf{p}}_m$ does not allow (3.52) and (3.54) to decouple into D scalar problems due to the form of $p_{\mathsf{y}\,|\,\mathbf{z}}$. However,

approximating $\boldsymbol{Q}_m^{\mathsf{p}}$ as diagonal and ignoring the off diagonal terms in $\boldsymbol{Q}_m^{\mathsf{z}}$ will make (3.52) and (3.54) easier to approximate.

There are three specific quantities we must compute to evaluate (3.52) and (3.54). *Note that we ignore the subscript m for brevity for the remainder of this section.* First, we must compute the scaling constant C, i.e.,

$$C = \int_{\boldsymbol{z}\in\mathbb{R}^D} p_{\mathsf{y}|\mathsf{z}}(y|\boldsymbol{z}) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathsf{p}})\, \mathrm{d}\boldsymbol{z}. \tag{3.55}$$

Second, the posterior mean of z_d, i.e.,

$$\widehat{z}_d = C^{-1} \int_{\boldsymbol{z}\in\mathbb{R}^D} z_d\, p_{\mathsf{y}|\mathsf{z}}(y|\boldsymbol{z}) \prod_{d'=1}^{D} \mathcal{N}(z_{d'}; \widehat{p}_{d'}, q_{d'}^{\mathsf{p}})\, \mathrm{d}\boldsymbol{z}. \tag{3.56}$$

And lastly, the posterior variance of z_d, i.e.,

$$q_d^{\mathsf{z}} = C^{-1} \int_{\boldsymbol{z}\in\mathbb{R}^D} z_d^2\, p_{\mathsf{y}|\mathsf{z}}(y|\boldsymbol{z}) \prod_{d'=1}^{D} \mathcal{N}(z_{d'}; \widehat{p}_{d'}, q_{d'}^{\mathsf{p}})\, \mathrm{d}\boldsymbol{z} - (\widehat{z}_d)^2. \tag{3.57}$$

The integrals in (3.55), (3.56) and (3.57) are intractable, so we have derived several approximate techniques to evaluate them, which we explain in the following subsections. Since all of the methods are only approximations, we will conclude this section with a performance evaluation where we test their accuracy and runtime.

Testing model Before presenting our methods of computing (3.55), (3.56) and (3.57), we will introduce our data generation model that we use for testing. Let our MMSE estimator be denoted by $\widehat{\boldsymbol{z}}_{\mathsf{mmse}} = f(y, \widehat{\boldsymbol{p}}, \boldsymbol{Q}^{\mathsf{p}})$. We generate data as follows.

1. Choose $\widehat{\boldsymbol{p}}$ and $\boldsymbol{Q}^{\mathsf{p}}$.
2. Generate $\boldsymbol{z}_{\mathsf{true}} \sim \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathsf{p}})$.
3. Generate $y_{\mathsf{true}} \sim p_{\mathsf{y}\,|\,\mathsf{z}}(y \,|\, \boldsymbol{z}_{\mathsf{true}})$.

4. Compute $\widehat{\boldsymbol{z}}_{\mathsf{mmse}} = f(y_{\mathsf{true}}, \widehat{\boldsymbol{p}}, \boldsymbol{Q}^{\mathsf{p}})$.

5. Compute the mean squared error (MSE) via $\text{MSE} = \|\boldsymbol{z}_{\mathsf{true}} - \widehat{\boldsymbol{z}}_{\mathsf{mmse}}\|_2^2$.

Since we are averaging over z and y, we are approximating the Bayes' Risk associated with the MMSE estimator in (3.56).

If we use the trivial estimator $\widehat{\boldsymbol{z}}_{\mathsf{triv}} = \widehat{\boldsymbol{p}}$ then our normalized MSE (defined by $\text{MSE}/q^{\mathsf{p}}$) is 1. We can use this as a baseline: if our estimator achieves a normalized MSE of more than one, then our estimator is doing more harm than good. In the sequel, $\widehat{\boldsymbol{p}} = [1, 0, 0, 0]^{\mathsf{T}}$, $\boldsymbol{Q}^{\mathsf{p}} = q^{\mathsf{p}} \boldsymbol{I}_D$ and $D = 4$. This represents a realistic and commonly occurring state within the MMSE-SHyGAMP algorithm.

Numerical integration

Just as in HyGAMP, the most basic approach to evaluating the integrals in (3.55), (3.56) and (3.57) is by numerical integration. As in MMSE-SHyGAMP, we use a D-dimensional hyper-rectangular grid where each dimension d has J grid points uniformly spaced between $[\widehat{p}_d - \alpha\sqrt{q_d^{\mathsf{p}}}, \widehat{p}_d + \alpha\sqrt{q_d^{\mathsf{p}}}]$. We evaluated the $\text{MSE}/q^{\mathsf{p}}$ over a grid of J and α in order to gain insight on appropriate parameter values. Based on the results shown in Figure 3.1, numerical integration should have an α of at least 3 and J greater than 6. The larger the scaling, the more points that are needed to ensure accuracy, however, the number of integration points ought to be kept as small as possible since this method requires J^D points for the integration.

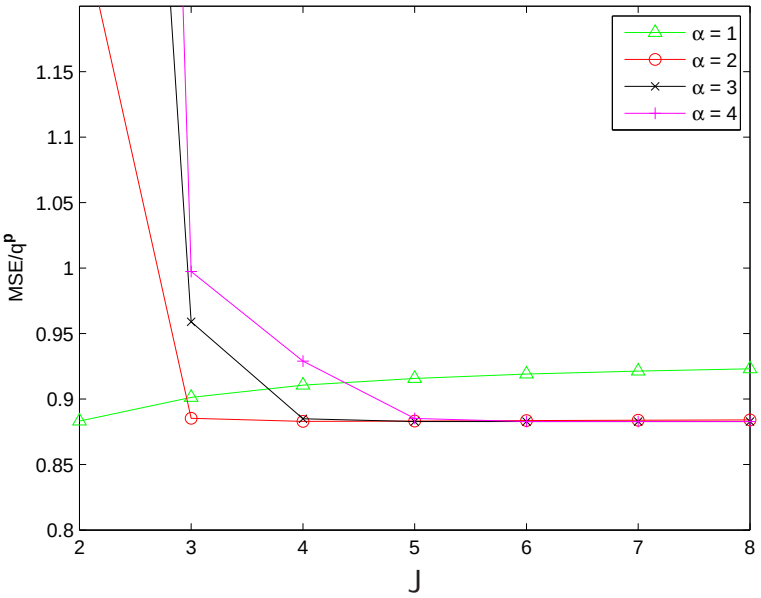

Figure 3.1: MSE/$q^{\mathbf{p}}$ vs the number of points per dimension J for different window sizes α of the estimator shown in (3.56) using the numerical integration method described in Section 3.3.2. Our test data is generated using the model described in Section 3.3.2 with $q^{\mathbf{p}} = 1$.

Importance sampling

We also revisit importance sampling for MMSE-SHyGAMP, which was previously described in Section 2.2.2. Here, we use a new sampling distribution of

$$p_{\mathbf{z}}(\widetilde{\boldsymbol{z}}) = \prod_{d=1}^{D} \mathcal{N}(\widetilde{z}_d; \widehat{p}_d, q_d^{\mathbf{p}}). \tag{3.58}$$

We evaluated this estimator's MSE/$q^{\mathbf{p}}$ versus the number of sample points J for various $q^{\mathbf{p}}$, as seen in Figure 3.2, and from this experiment we determined $J = 1500$ was an appropriate default.

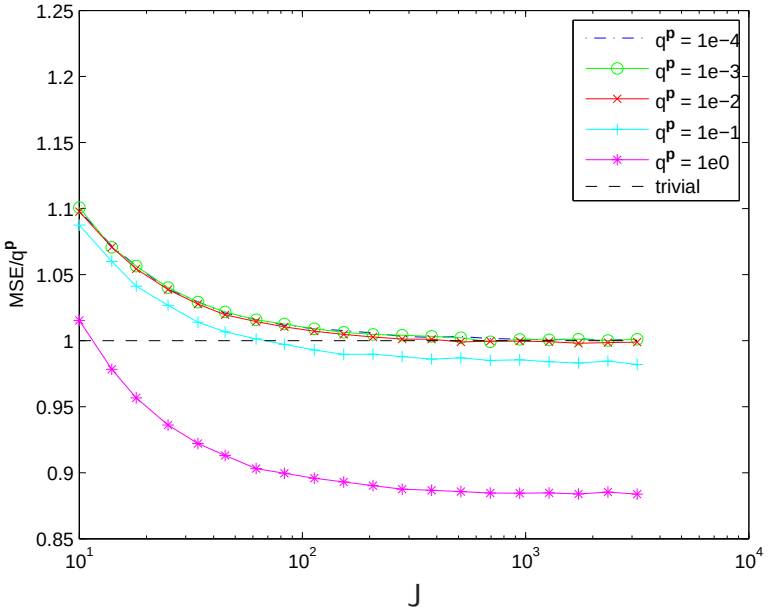

Figure 3.2: MSE/$q^{\mathbf{p}}$ vs J for various $q^{\mathbf{p}}$ using the importance sampling method described in Section 3.3.2. Our test data is generated using the model described in Section 3.3.2.

Taylor series approximation

In a different approach to computing (3.55), (3.56), and (3.57), we approximate the multinomial logistic likelihood with a second order Taylor series about the point $\boldsymbol{z} = \widehat{\boldsymbol{p}}$, which leads to closed-form solutions for the (approximate) posterior mean and variance. We note that this method breaks down when elements in $\boldsymbol{Q}^{\mathbf{p}} \gtrsim 10$ because the quadratic approximation to the multinomial logistic likelihood becomes poor away from $\widehat{\boldsymbol{p}}$. Let $f(\boldsymbol{z}) \triangleq p_{\mathsf{y}\,|\,\mathbf{z}}(y\,|\,\boldsymbol{z})$. After the Taylor series approximation, the

general form of the posterior is

$$p_{\mathbf{z}\,|\,\mathbf{y},\mathbf{p}}(\boldsymbol{z}\,|\,y,\widehat{\boldsymbol{p}};\boldsymbol{Q}^{\mathbf{p}}) \propto \left(f(\widehat{\boldsymbol{p}}) + \sum_{j=1}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j}(z_j - \widehat{p}_j) \right.$$
$$\left. + \frac{1}{2}\sum_{j=1}^{D}\sum_{k=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_j \partial z_k}(z_j - \widehat{p}_j)(z_k - \widehat{p}_k) \right) \times \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}), \quad (3.59)$$

where the first factor comes from the Taylor approximation to the likelihood and the second factor is the prior distribution. The normalizing constant is then approximated by

$$C \approx f(\widehat{\boldsymbol{p}}) + \frac{1}{2}\sum_{d=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_d^2} q_d^{\mathbf{p}}. \quad (3.60)$$

The approximate posterior mean is given by

$$\widehat{z}_i \approx C^{-1} \left(f(\widehat{\boldsymbol{p}})\,\widehat{p}_i + \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i}(q_i^{\mathbf{p}}) + \frac{1}{2}\sum_{j=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_j}^2}(\widehat{p}_i q_j^{\mathbf{p}}) \right), \quad (3.61)$$

and the approximate variance is given by

$$q_i^{\mathbf{z}} \approx C^{-1} \left(f(\widehat{\boldsymbol{p}})\,(\widehat{p}_i^2 + q_i^{\mathbf{p}}) + 2\frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i}\widehat{p}_i q_i^{\mathbf{p}} + \frac{1}{2}\frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_i}^2} q_i^{\mathbf{p}}(3q_i^{\mathbf{p}} + \widehat{p}_i^2) \right.$$
$$\left. + \frac{1}{2}\frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_i}^2}(\widehat{p}_i^2 + q_i^{\mathbf{p}}) \sum_{j=1,\neq i}^{D} q_j^{\mathbf{p}} \right) - \widehat{z}_i^2. \quad (3.62)$$

The derivations for (3.60), (3.61) and (3.62) are in Appendix B.1.

Gaussian posterior approximation

In another method to approximate $\widehat{\boldsymbol{z}}$ and $\boldsymbol{Q}^{\mathbf{z}}$, we approximate (2.4) with a Gaussian distribution. We aim to write

$$\log p_{\mathbf{z}\,|\,\mathbf{y},\mathbf{p}}(\boldsymbol{z}\,|\,y,\widehat{\boldsymbol{p}};\boldsymbol{Q}^{\mathbf{p}}) \approx \frac{1}{2}\boldsymbol{z}^T\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}^T\boldsymbol{z} + c \quad (3.63)$$
$$= \frac{1}{2}(\boldsymbol{z} + \boldsymbol{W}^{-1}\boldsymbol{b})^T\boldsymbol{W}(\boldsymbol{z} + \boldsymbol{W}^{-1}\boldsymbol{b}) + c - \frac{1}{2}\boldsymbol{b}^T\boldsymbol{W}\boldsymbol{b}, \quad (3.64)$$

with negative definite $\boldsymbol{W}$. The approximate posterior mean vector is given by $\widehat{\boldsymbol{z}} = -\boldsymbol{W}^{-1}\boldsymbol{b}$ and the approximate posterior covariance matrix is given by $\boldsymbol{Q}^{\mathbf{z}} = [-\boldsymbol{W}]^{-1}$.

The log of the posterior is

$$\log p_{\mathbf{z}\,|\,\mathrm{y},\mathbf{p}}(\boldsymbol{z}\,|\,y,\widehat{\boldsymbol{p}};\boldsymbol{Q}^{\mathbf{p}}) = \\ \left(\sum_{d=1}^{D}\left(-\frac{1}{2q_d^{\mathbf{p}}}(z_d-\widehat{p}_d)^2+\frac{1}{\sqrt{2\pi q_d^{\mathbf{p}}}}\right)+\log\frac{\exp(z_y)}{\sum_{d=1}^{D}\exp(z_d)}\right). \tag{3.65}$$

Notice all of the terms are already quadratic except for the last term, the log likelihood. After approximating $f(\boldsymbol{z}) \triangleq \log\left(\frac{\exp(z_y)}{\sum_{d=1}^{D}\exp(z_d))}\right)$ with a second order Taylor series about the point $\widehat{\boldsymbol{p}}$, elements of $\boldsymbol{W}$ in are given by

$$W_{ij} = 2\frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_i \partial z_j} \text{ if } i \neq j$$
$$W_{ij} = 2\frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_i \partial z_j} - \frac{1}{q_i^{\mathbf{p}}} \text{ if } i = j,$$

and elements of $\boldsymbol{b}$ in (3.63) are given by

$$b_i = \frac{\widehat{p}_i}{q_i^{\mathbf{p}}} - \sum_{j=1}^{D}\widehat{p}_j\frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_i \partial z_j} + \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i}.$$

The derivations for $\boldsymbol{W}$ and $\boldsymbol{b}$ can be found in Appendix B.2. Similar to the Taylor series approximation method, this method breaks down when elements in $\boldsymbol{Q}^{\mathbf{p}}$ are relatively large.

Gaussian-mixture approximation

The final method we tried for evaluating (3.55), (3.56), and (3.57) is similar to the Taylor series method but is based on Gaussian-mixtures (GM). It is known that the logistic cdf $1/(1+\exp(-x))$ is well approximated by a mixture of a few Gaussian cdfs, which leads to an efficient method of approximating (3.52)-(3.54) in the case of *binary* logistic regression (i.e., $D = 2$) [27]. 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

$$\delta_d^{(y)} \triangleq \begin{cases} z_y - z_d & d \neq y \\ z_y & d = y \end{cases}. \tag{3.66}$$

Their utility can be seen from the fact that (recalling (1.5))

$$p_{\mathsf{y}|\mathbf{z}}(y|\boldsymbol{z}) = \frac{1}{1 + \sum_{d \neq y} \exp(z_d - z_y)} \tag{3.67}$$

$$= \frac{1}{1 + \sum_{d \neq y} \exp(-\delta_d^{(y)})} \triangleq l^{(y)}(\boldsymbol{\delta}^{(y)}), \tag{3.68}$$

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

$$l^{(y)}(\boldsymbol{\delta}) \approx \sum_{l=1}^{L} \alpha_l \prod_{k \neq y} \Phi\left(\frac{\delta_k - \mu_{kl}}{\sigma_{kl}}\right) \triangleq \widehat{l}^{(y)}(\boldsymbol{\delta}), \tag{3.69}$$

where $\sigma_{kl}^{(k)} > 0$, $\alpha_l \geq 0$, 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 $\sup_{\boldsymbol{\delta} \in \mathbb{R}^D} |l^{(y)}(\boldsymbol{\delta}) - \widehat{l}^{(y)}(\boldsymbol{\delta})|$.

Recall from (3.55)-(3.57) that our objective is to compute quantities of the form

$$\int_{\mathbb{R}^D} (\boldsymbol{e}_d^{\mathsf{T}} \boldsymbol{z})^i \, p_{\mathsf{y}|\mathbf{z}}(y|\boldsymbol{z}) \mathcal{N}(\boldsymbol{z}; \widehat{\boldsymbol{p}}, \boldsymbol{Q}^{\mathsf{p}}) \, \mathrm{d}\boldsymbol{z} \triangleq S_{di}^{(y)} \tag{3.70}$$

where $i \in \{0, 1, 2\}$ and $\boldsymbol{e}_d$ is the dth column of $\boldsymbol{I}_D$. To exploit (3.69), we change the variable of integration to

$$\boldsymbol{\delta}^{(y)} = \boldsymbol{T}_y \boldsymbol{z} \tag{3.71}$$

with

$$\boldsymbol{T}_y = \begin{bmatrix} -\boldsymbol{I}_{y-1} & \mathbf{1}_{(y-1)\times 1} & \mathbf{0}_{(y-1)\times(D-y)} \\ \mathbf{0}_{1\times(y-1)} & 1 & \mathbf{0}_{1\times(D-y)} \\ \mathbf{0}_{(D-y)\times(y-1)} & \mathbf{1}_{(D-y)\times 1} & -\boldsymbol{I}_{D-y} \end{bmatrix} \tag{3.72}$$

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

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

$$S_{di}^{(y)} = \int_{\mathbb{R}^D} \left(\boldsymbol{e}_d^\mathsf{T} \boldsymbol{T}_y^{-1} \boldsymbol{\delta}\right)^i l^{(y)}(\boldsymbol{\delta}) \mathcal{N}\left(\boldsymbol{\delta}; \boldsymbol{T}_y \widehat{\boldsymbol{p}}, \boldsymbol{T}_y \boldsymbol{Q}^{\mathbf{p}} \boldsymbol{T}_y^\mathsf{T}\right) \mathrm{d}\boldsymbol{\delta}. \tag{3.73}$$

Then, applying the approximation (3.69) and

$$\mathcal{N}\left(\boldsymbol{\delta}; \boldsymbol{T}_y \widehat{\boldsymbol{p}}, \boldsymbol{T}_y \boldsymbol{Q}^{\mathbf{p}} \boldsymbol{T}_y^\mathsf{T}\right) = \mathcal{N}\left(\delta_y; \widehat{p}_y, q_y^{\mathbf{p}}\right) \times \prod_{k \neq y} \mathcal{N}\left(\delta_k; \delta_y - \widehat{p}_k, q_k^{\mathbf{p}}\right) \tag{3.74}$$

to (3.73), we find that

$$\begin{aligned} S_{di}^{(y)} \approx \sum_{l=1}^{L} \alpha_l \int_{\mathbb{R}} \mathcal{N}\left(\delta_y; \widehat{p}_y, q_y^{\mathbf{p}}\right) \Bigg[\int_{\mathbb{R}^{D-1}} \left(\boldsymbol{e}_d^\mathsf{T} \boldsymbol{T}_y^{-1} \boldsymbol{\delta}\right)^i \\ \times \prod_{k \neq y} \mathcal{N}\left(\delta_k; \delta_y - \widehat{p}_k, q_k^{\mathbf{p}}\right) \Phi\left(\frac{\delta_k - \mu_{kl}}{\sigma_{kl}}\right) \mathrm{d}\delta_k \Bigg] \mathrm{d}\delta_y. \end{aligned} \tag{3.75}$$

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

$$\boldsymbol{e}_d^\mathsf{T} \boldsymbol{T}_y^{-1} \boldsymbol{\delta} = \begin{cases} \delta_y - \delta_d & d \neq y \\ \delta_y & d = y \end{cases}. \tag{3.76}$$

Thus, for a fixed value of $\delta_y = v$, the inner integral in (3.75) can be expressed as a linear combination of terms like

$$\int_{\mathbb{R}} \delta^i \mathcal{N}\left(\delta; v - \widehat{p}, q\right) \Phi\left(\frac{\delta - \mu}{\sigma}\right) \mathrm{d}\delta \triangleq T_i \tag{3.77}$$

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

$$T_0 = \Phi(x) \tag{3.78}$$

$$T_1 = (v - \widehat{p})\Phi(x) + \frac{q\phi(x)}{\sqrt{\sigma^2 + q}} \tag{3.79}$$

$$T_2 = \frac{(T_1)^2}{\Phi(x)} + q\Phi(x) - \frac{q^2 \phi(x)}{\sigma^2 + q}\left(x + \frac{\phi(x)}{\Phi(x)}\right) \tag{3.80}$$

which can be obtained using the results in [28, §3.9]. The outer integral in (3.75) can then be approximated via numerical integration over a grid of J values. Next, we will determine appropriate values of L and J.

Figure 3.3 shows the GM method's MSE for one through four mixture components when the number of integration points is fixed at seven. It can be seen that going from one to two mixture components offers a large decreases in MSE, while two to three is a modest decrease in MSE and three to four is almost no change in MSE. Figure 3.4 shows the estimator MSE for various numbers of numerical integration points when the number of mixture components is fixed at two. We see decreases in MSE up until approximately $J = 6$. Figure 3.5 shows the effect of the number of mixture components on runtime. The runtime increases linearly with the number of mixture components, however the marginal increase in runtime is small. Figure 3.6 shows the effect on runtime as the number of numerical integration points increases. Similar to the effect on runtime versus the number of mixture components, the runtime increases linearly but the marginal increase is small.

Estimator performance comparison

We have now shown five different methods for computing the output estimator in MMSE-SHyGAMP. In this section we will evaluate the accuracy and runtime of each method.

Table 3.1 shows the default parameters for each method. To test the performance of the estimators with their default parameters, each estimator implementation method's MSE was evaluated using the procedure described in Section 3.3.2, for values of q^{p} between 10^{-3} and 10^{3}. The results of this test are in Figure 3.7.

Each data point in both experiments is an average of 5e6 trials. We can see that the Taylor series and Gaussian posterior approximation methods break down when q^{p} is large. Furthermore, the numerical integration, importance sampling, and

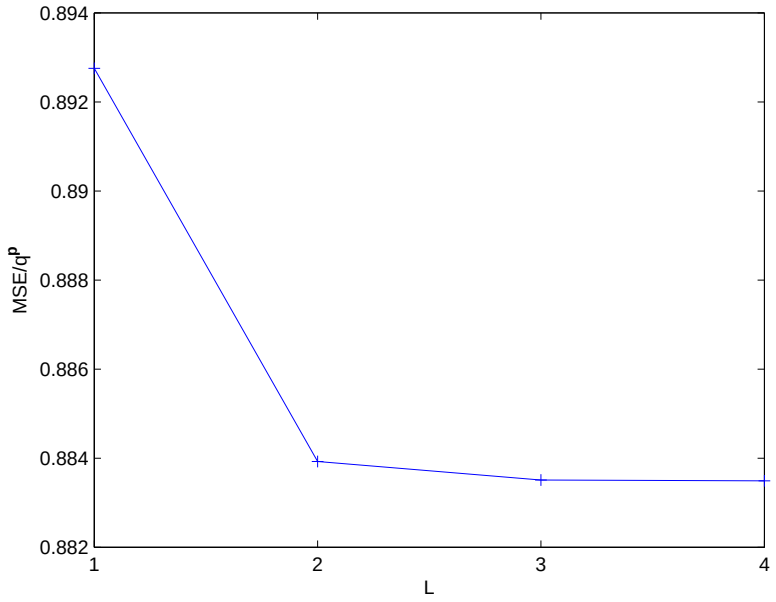

Figure 3.3: MSE/$q^{\mathbf{p}}$ versus the number of GM Components L. Our test data is generated using the model described in Section 3.3.2. $q^{\mathbf{p}} = 1$.

Gaussian-mixture methods are tightly bundled, but the Gaussian-mixture method is *slightly* better for all $q^{\mathbf{p}}$, so we will look to runtime to see which method is the best.

Figure 3.8 plots average runtime, over 2000 trials and for fixed $M = 500$, versus D for the different MMSE estimator implementation methods. From this figure, we see the Taylor series method is the fastest, followed by the Gaussian-mixture method. Since the Taylor series method is not robust to $q^{\mathbf{p}}$, and the Gaussian-mixture method is accurate for all $q^{\mathbf{p}}$, we use the Gaussian-mixture method as our default for the MMSE output estimator.

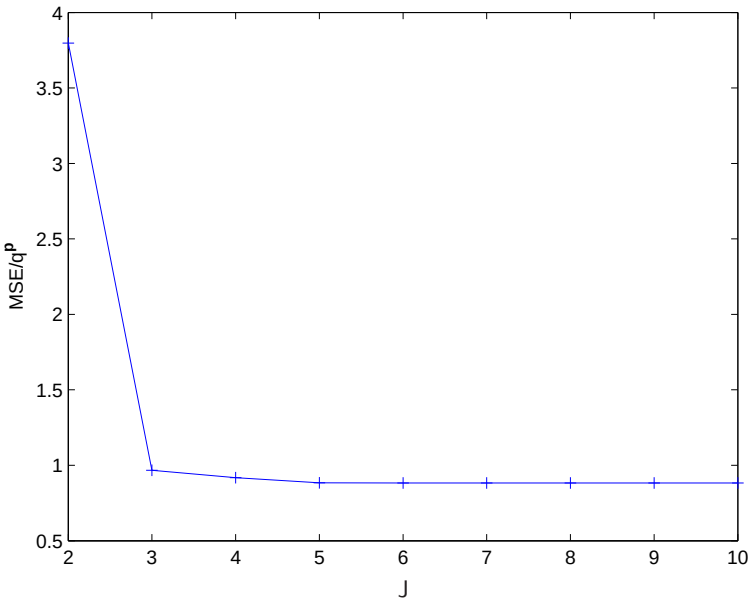

Figure 3.4: MSE/$q^{\mathbf{p}}$ versus the number of numerical integration points J in the Gaussian mixture method. Our test data is generated using the model described in Section 3.3.2. $q^{\mathbf{p}} = 1$.

3.3.3 Parameter selection for MMSE-SHyGAMP

Recall, in MMSE-SHyGAMP, we propose to use the Bernoulli-Gaussian prior in (3.48), which has parameters λ_0, μ_0, and σ_0^2. Rather than use cross-validation to tune these parameters, we use the EM-GM-AMP framework described in [29] to tune the parameters online[2].

The parameters λ_0 and σ_0 require careful initialization for the EM learning procedure to work. We use the following approximate inequality to help set λ_0:

$$M \log_2 D \gtrsim K\, D \log_2 \left(\frac{N}{K} \right), \tag{3.81}$$

[2]We note that we choose to fix $\mu_0 = 0$.

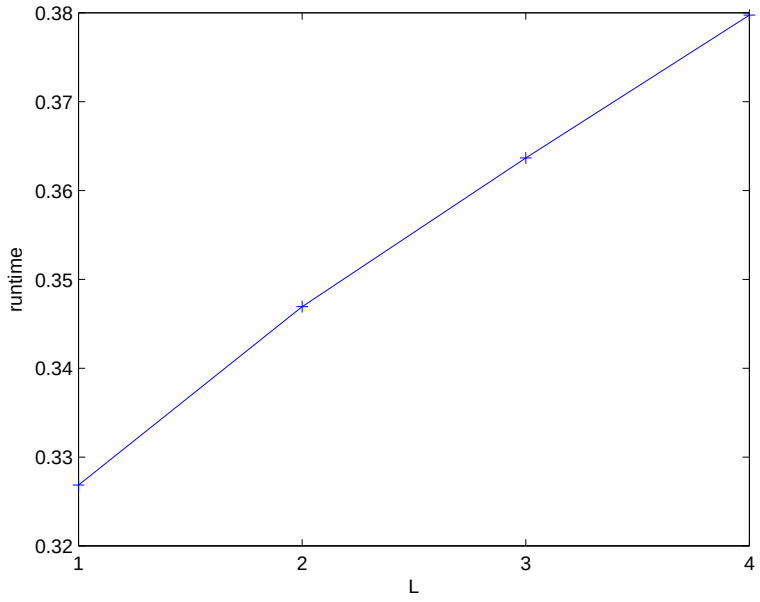

Figure 3.5: Cumulative runtime versus the number of GM components L for 1000 trials.

where $\lambda_0 = \frac{K}{N}$. Note that here, K is referring to the number of non-zero rows of the "true" weight-matrix, and in practice is unknown. The left side of (3.81) is the entropy of the training data; the labels contain $\log_2 D$ bits of information since we assume the class labels have a uniform distribution and there are M of them, hence $M \log_2 D$. The right hand side of (3.81) comes from combinatorics. A simplified model is to assume each weight vector contains K non-zero entries, leading to $\binom{N}{K}$ combinations in each of the D weight vectors, which is lower-bounded by $\left(\frac{N}{K}\right)^K$. Since there are D weight vectors, a lower bound on the total entropy is $K\, D \log_2 \left(\frac{N}{K}\right)$, hence the right side of (3.81). M, N and D are known, so we will choose K_0 as one less than the smallest value of K where (3.81) doesn't hold and we require $K_0 \leq N$. Then, we will set $\lambda_0 = \min\{\frac{M}{N}, 1\}$ and $\sigma_0^2 = \frac{\widehat{c}^2}{N\lambda_0(\widehat{\sigma}_a^2)^2}$, where $\widehat{c}$ and $\widehat{\sigma}_a^2$ correspond to the norm of

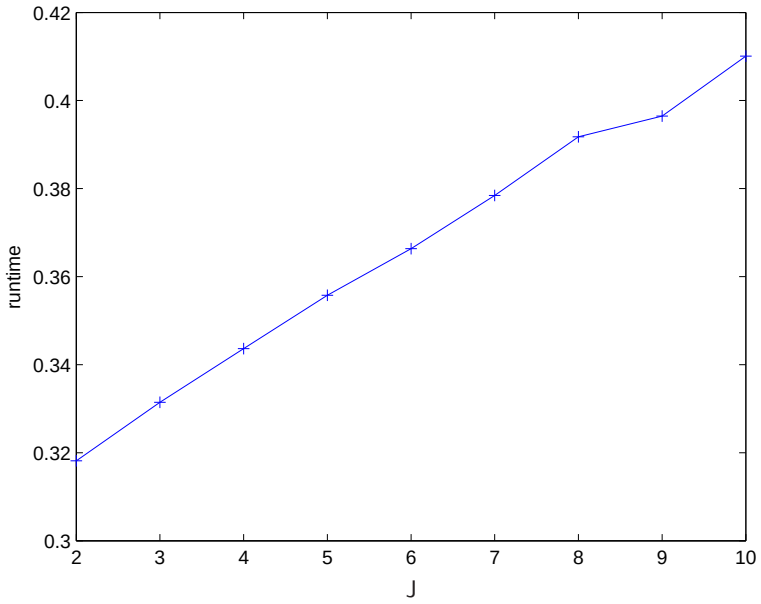

Figure 3.6: Total runtime versus the number of numerical integration points J for 1000 trials.

the mean and cloud variance of each class (see the description of our data model in Section 1.3.1), and are estimated from the training data.

3.4 Conclusion

In this Chapter we designed two new algorithms based on simplifications to the Hybrid-GAMP algorithms for MAP and MMSE estimation. In particular, we approximated all covariance matrices as diagonal. This in part led to MAP-SHyGAMP requiring a less complex implementation of the sparsity-promoting Laplacian prior. Furthermore, the diagonal covariance approximation combined with a scalar variance approximation led to a SURE-based online parameter tuning technique for the Laplacian prior in MAP-SHyGAMP.

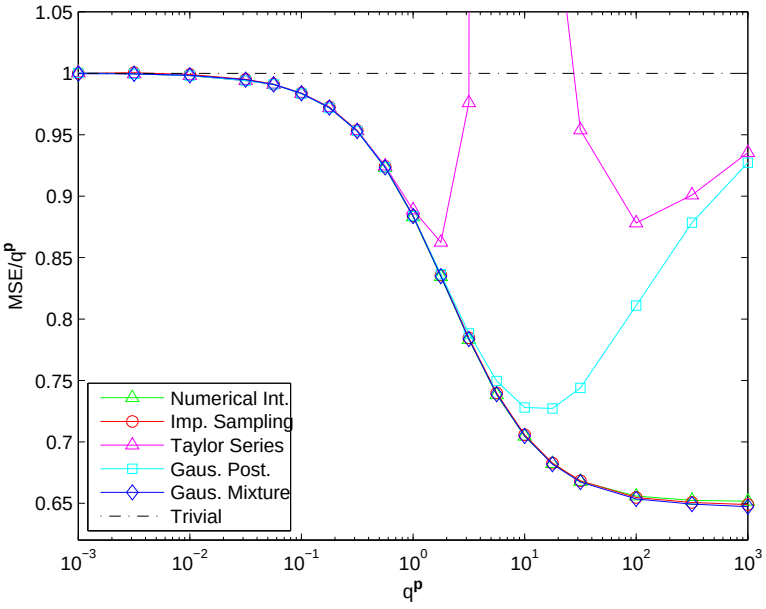

Figure 3.7: MSE/$q^{\mathbf{p}}$ vs $q^{\mathbf{p}}$ of the estimator shown in (3.56) using the implementation methods described in Section 3.3.2. The default parameters for each implementation method (shown in Table 3.1) were used. The data was created with the standard testing model described in Section 3.3.2. Each data point is an average of 5e6 trials.

Description	Parameters	Default Value
Numerical Integration	α - scaling J - grid size	4 7
Importance Sampling	J - number of samples	1500
Taylor Series Approximation to (2.5)	n/a	n/a
Gaussian Approximation to (2.4)	n/a	n/a
Gaussian Mixture Approximation to (3.68)	L - number of mixture components J - grid size	2 7

Table 3.1: Summary of the default parameters of the estimator implementation techniques in MMSE-SHyGAMP.

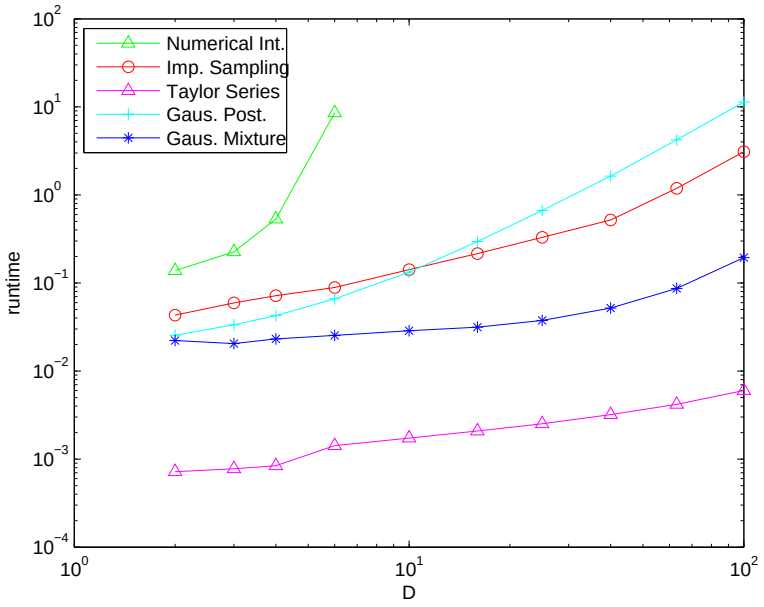

Figure 3.8: Cumulative runtime of 500 realizations of each MMSE output estimation method versus the dimensionality of the variables, D. Each point is an average of $T = 2000$ trials.

In MMSE-SHyGAMP, we developed a Gaussian-mixture approach to implementing the output estimator that was as accurate as other approaches but less computationally complex. Also, in MMSE-SHyGAMP, we tuned the parameters associated with the Bernoulli-Gaussian prior online using EM.

Chapter 4: Classification Experiments

In this chapter we show how well our SHyGAMP algorithms perform relative to two state-of-the-art sparse multinomial logistic regression algorithms: GLMNET [15] and SBMLR [16]. Our performance metrics are classification error rate and runtime, and we test the algorithms on both synthetic and real data. In all of our experiments, we used ℓ_1 regularization in MAP-SHyGAMP, and the Bernoulli-Gaussian prior in MMSE-SHyGAMP. Both MAP and MMSE-SHyGAMP employed online parameter tuning, as described in Section 3.2.3 and Section 3.3.3. GLMNET also used ℓ_1 regularization, but the weight of the regularizing term must be tuned via cross-validation; we used 10-fold cross-validation over a grid of 100 lambda values. SBMLR can tune its parameters online, thus did not require cross-validation. The runtimes we report include the time spent performing cross-validation (if applicable).

4.1 Synthetic data

Data generation model In our synthetic data experiments, our data was generated according to the model specified in Section 1.3.1. Specifically, we generated D different length-N, K-sparse, random, orthogonal class means of norm c whose nonzero entries were determined by the $K \times K$ unitary matrix, $\boldsymbol{U}$, scaled by c. $\boldsymbol{U}$ was generated by taking the singular value decomposition of an i.i.d. Gaussian random

matrix. We note that each class mean had the same support. Finally, Gaussian noise with covariance matrix $\sigma_a^2 \boldsymbol{I}$ was added to the means generated from the previous steps to obtain the rows of the feature matrix $\boldsymbol{A}$. The parameters σ_a^2 and c were chosen to meet a particular Bayes' error rate (BER), based on (1.27).

Effects of M, N, K on performance We performed three synthetic data simulations, where in each simulation one parameter of the data was varied while the rest remained fixed. In the first simulation, we fixed $D = 4$, $N = 10000$, and $K = 10$ and varied M from 100 to 5000. The classification error rates of the different algorithms are shown in Fig. 4.1 and their runtimes are shown in Fig. 4.2. From these plots, we can see that MMSE-SHyGAMP won in error, followed by MAP-SHyGAMP. Also, MMSE-SHyGAMP and MAP-SHyGAMP won in runtime (their times are very close) as M grew large.

In the second simulation, we fixed $D = 4$, $M = 200$, and $K = 10$ while we varied N from 10^3 to $10^{5.5} \approx 320000$. The classification error rates of the different algorithms are shown in Fig. 4.3 and their runtimes are shown in Fig. 4.4. From these plots, we can see that MAP-SHyGAMP always beat SBMLR and GLMNET in error, while MMSE-SHyGAMP sometimes beat MAP-SHyGAMP. Also, for large N, MAP-SHyGAMP was generally the fastest algorithm, only occasionally losing to MMSE-SHyGAMP.

In the third and final simulation, we fixed $D = 4$, $M = 300$, and $N = 30000$ while we varied K from 5 to 30. The classification error rates of the different algorithms are shown in Fig. 4.5 and their runtimes are shown in Fig. 4.6. From these plots, we can see that MMSE-SHyGAMP won in error, followed by MAP-SHyGAMP. Also,

MAP-SHyGAMP won in runtime, followed by MMSE-SHyGAMP. Lastly, all of the algorithms' runtimes were reasonably invariant to K.

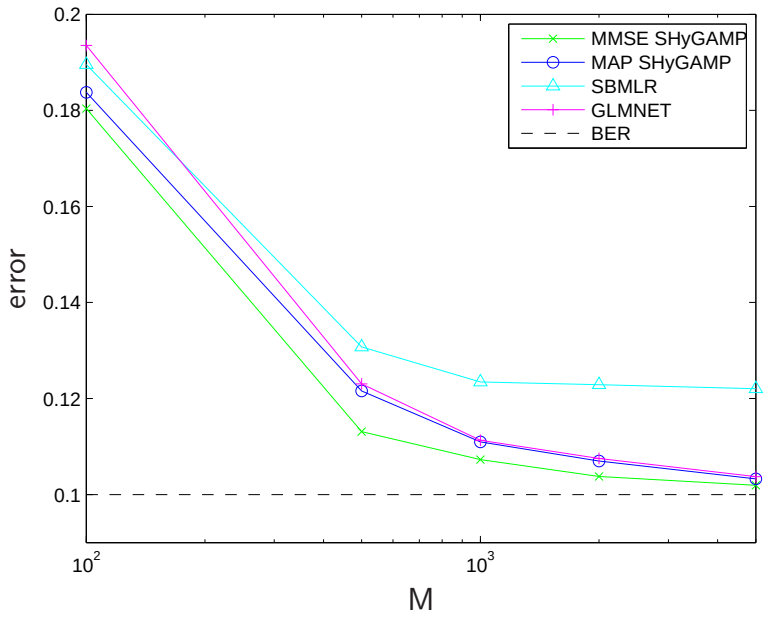

Figure 4.1: Classification error rate vs M for $D = 4$, $N = 10000$, and $K = 10$. Each data point is an average of 12 trials.

4.2 fMRI Multi-Voxel Pattern Analysis

The next experiment was performed on the Haxby dataset [2]. This dataset consists of feature/label pairs where the features are fMRI brain images (with $N = 31398$ voxels per image) and the labels are cognitive tasks. The goal is to perform feature selection, i.e., determine which areas if the brain can be used to predict each cognitive task. In this application, classification error rate can be used as a metric to determine if the learned brain-areas yield accurate task predictions.

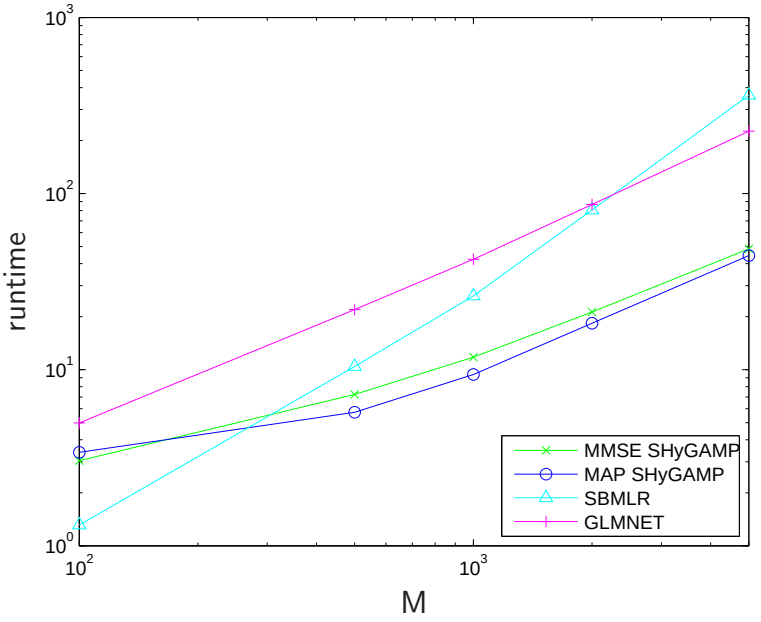

Figure 4.2: Algorithm runtime vs M for $D = 4$, $N = 10000$, and $K = 10$. Each data point is an average of 12 trials.

Each task corresponds to the human subject looking at a particular class of object (e.g., cat, house, chair). The dataset includes 120 fMRI images per object class, for a maximum of $M = 840$. We wish to test our classification algorithms on this dataset because the number of potentially discriminating features far outnumbers the number of training samples.

In our experiment, for a particular subset of classes, we performed 25 trials, where in each trial we selected 10% of the data at random to be used as test data and used the other 90% as training data. The average test-error-rate and runtime for two different subsets of classes are shown in Table 4.1. From this table, we can see that GLMNET was best w.r.t. error, followed by SBMLR, then MAP-SHyGAMP, and lastly MMSE-SHyGAMP. However, we urge caution in interpreting the results in

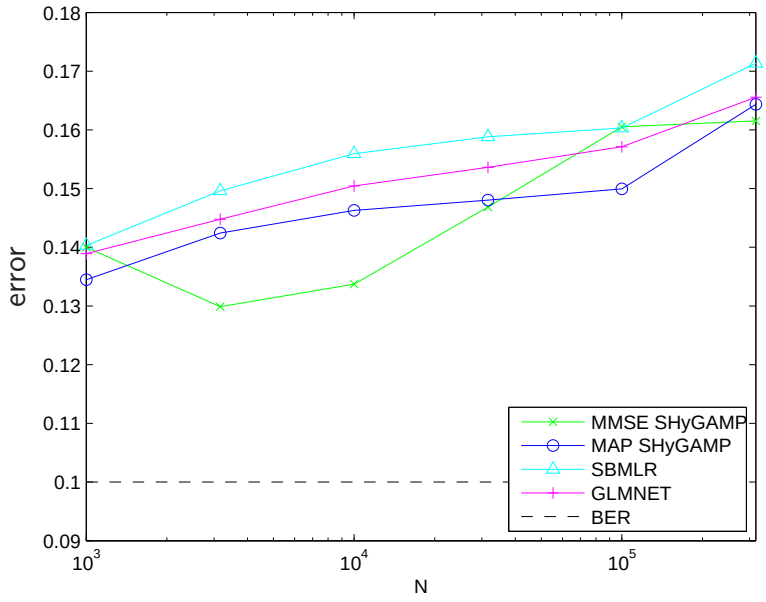

Figure 4.3: Classification error rate vs N for $D = 4$, $M = 200$, and $K = 10$. Each data point is an average of 12 trials.

Table 4.1. The number of test samples is very few, and so the test-error-rate estimate has a very high variance. In any case, we do note that the runtimes of MMSE and MAP SHyGAMP were much faster than those of the competitors.

4.3 Text mining

Our next experiment is intended to test our algorithms on a dataset which is very high dimensional, has a large number of classes, and a large number of training samples. The RCV1 dataset [4] provides us with this opportunity. In this dataset, classes are news article categories and features are word frequency per article. We used a subset of the first 25 classes of the total dataset, with dimensions $N = 47236$, $M_{\text{train}} = 14147$, $M_{\text{test}} = 469571$, and $D = 25$.

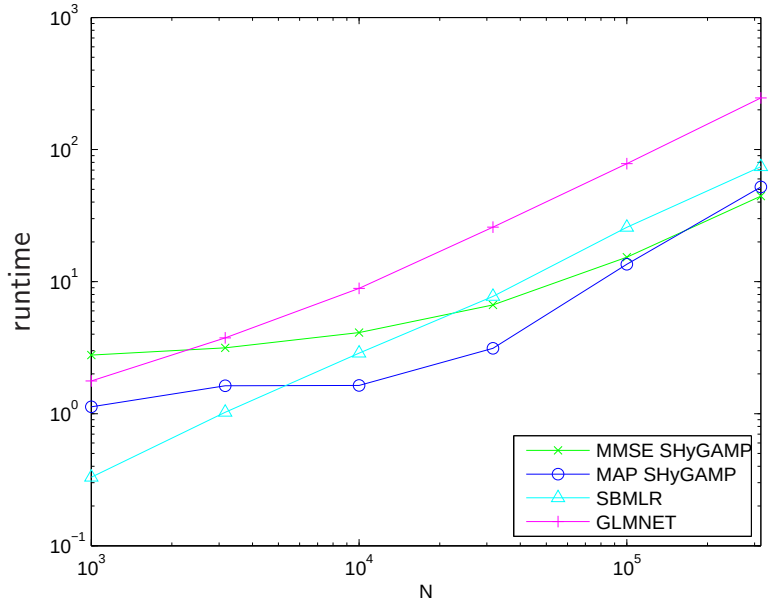

Figure 4.4: Algorithm runtime vs N for $D = 4$, $M = 200$, and $K = 10$. Each data point is an average of 12 trials.

This dataset is sparse and has non-zero-mean features, which provides an additional complication for algorithms such as SHyGAMP, which assume approximately i.i.d. zero-mean $\boldsymbol{A}$. For example, due to memory limitations we could not z-score the training data. However, other algorithms had even more difficulty. For example, GLMNET diverged, which we conjecture was due to the sparsity in the feature matrix.

In this experiment, we looked at the test error rate vs runtime for MMSE-SHyGAMP, MAP-SHyGAMP and SBMLR. We did not run GLMNET on this dataset. For the algorithms under test, we plot their test error rate vs runtime in Fig. 4.7. From this plot, we can see that MAP-SHyGAMP converged the fastest, followed by MMSE-SHyGAMP, and SBMLR was the slowest.

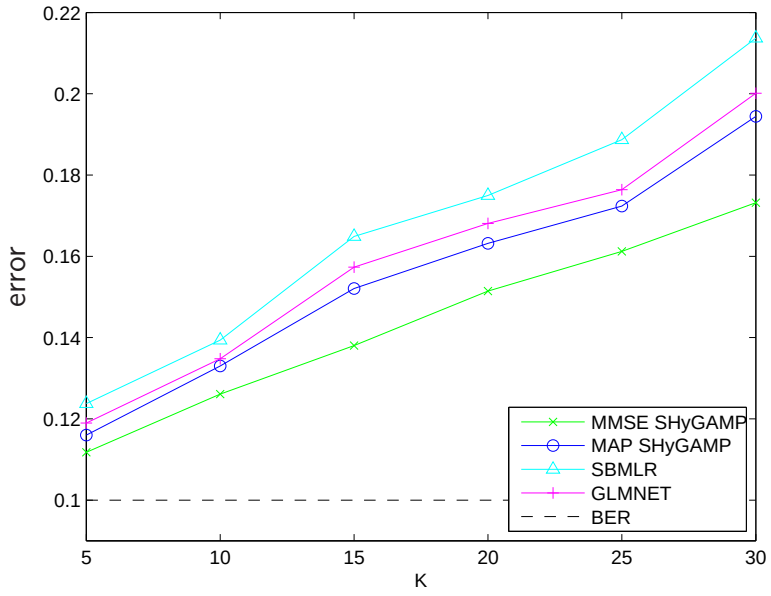

Figure 4.5: Classification error rate vs K for $D = 4$, $M = 300$, and $N = 30000$. Each data point is an average of 12 trials.

4.4 Micro-array gene expression

Next, we looked at micro-array gene expression data. This dataset was taken from [5] and consists of $M = 180$ samples each consisting of $N = 54613$ micro-array gene expression values. The samples are split into $D = 4$ different classes: one control class and three classes representing different types of glioma.

Our classification experiment on this dataset consisted of 100 trials where in each trial we held out 25% of the data at random to be used for testing and used the other 75% as training data. The average classification error rates, runtimes, and estimated sparsities over the 100 trials are shown in Table 4.2. From this data, we see MMSE-SHyGAMP was the fastest, but the worst in error. MAP-SHyGAMP

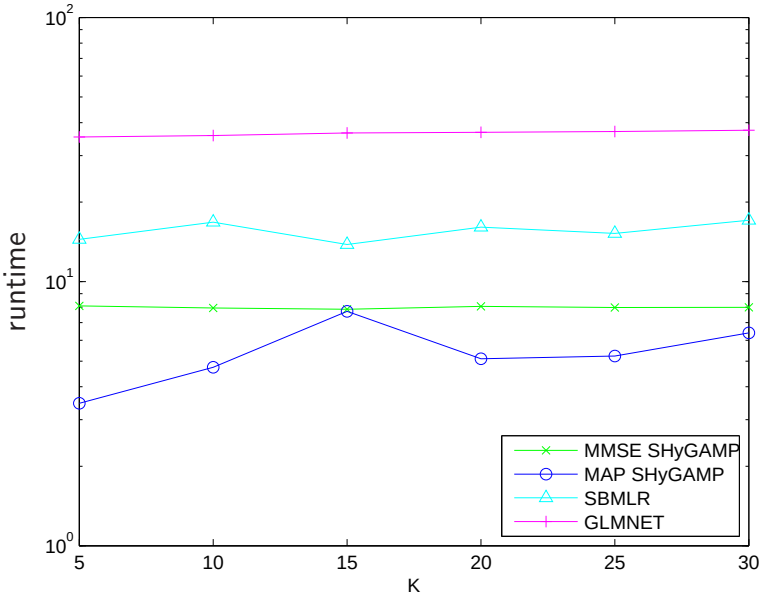

Figure 4.6: Algorithm runtime vs K for $D = 4$, $M = 300$, and $N = 30000$. Each data point is an average of 12 trials.

was the second fastest and the best in error. GLMNET learned a sparser solution than MAP-SHyGAMP, and MAP-SHyGAMP and SBMLR agreed very closely in estimated sparsity. However, we urge caution in interpreting the results in Table 4.2. The number of test samples is very few, and so the test-error-rate estimate has a very high variance (note the test-error-rate standard deviation in Table 4.2).

4.5 Conclusion

In conclusion, we have demonstrated that MMSE and MAP-SHyGAMP are competitive with the algorithms SBMLR and GLMNET w.r.t. both error and runtime, on both synthetic and real data. Furthermore, we have seen that the online parameter

	Five Class		Seven Class	
Algorithm	Error (SD) %	Runtime (s)	Error (SD) %	Runtime (s)
MMSE SHyGAMP	9.8 (4.5)	12.35	16.2 (4.6)	16.99
MAP SHyGAMP	4.9 (3.1)	18.95	10.9 (3.2)	26.93
SBMLR	4.4 (2.9)	33.90	10.1 (2.6)	83.83
GLMNET	2.7 (2.3)	93.48	6.7 (3.0)	213.78

Table 4.1: Average classification results and runtimes on the Haxby dataset, for 25 trials. The number in parenthesis is the standard deviation of the test-error-rate. The classes used in the five-class case were 'cat', 'house', 'shoe', 'face', and 'chair'. The classes used in the seven-class case were 'cat', 'house', 'shoe', 'face', 'chair', 'scissors', and 'bottle'.

Algorithm	Error (SD) (%)	Runtime (s)	$\widehat{K}_{99}$	$\widehat{K}_{\ell 0}$
MMSE SHyGAMP	34.0 (7.1)	8.36	10.60	218 452.00
MAP SHyGAMP	31.5 (7.1)	14.39	36.94	57.86
SBMLR	31.8 (6.9)	20.69	39.39	58.79
GLMNET	32.5 (6.5)	29.56	23.17	35.48

Table 4.2: Classification results, runtimes, and estimated sparsities on the Sun dataset. The number in parenthesis indicates the standard deviation of the test-error-rate. Each data point is an average of 100 trials. $\widehat{K}_{99}$ indicates the estimated sparsity by taking the number of weights which make up 99% of the Frobenius norm of $\widehat{\boldsymbol{X}}$. $\widehat{K}_{\ell 0}$ is the number of non-zero coefficients in $\widehat{\boldsymbol{X}}$.

tuning methods employed by MMSE and MAP-SHyGAMP are both computationally efficient and effective at minimizing the test error rate.

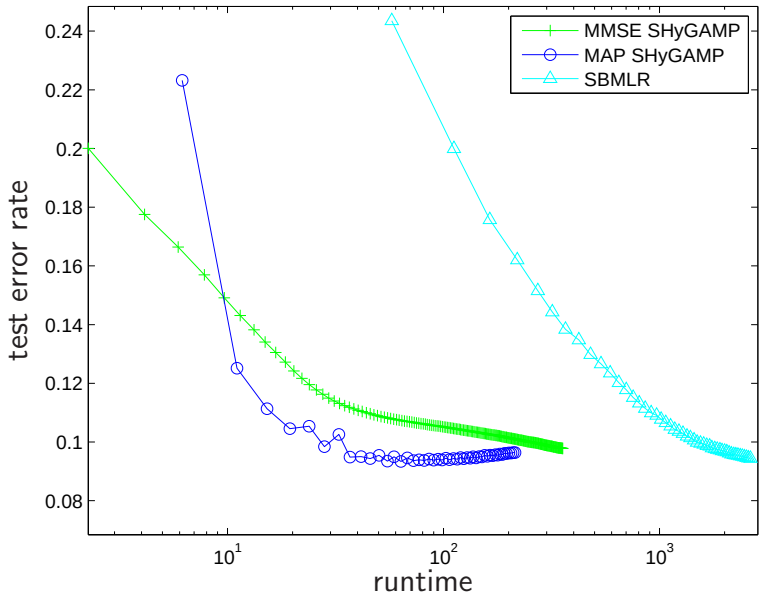

Figure 4.7: Test-error-rate vs runtime on the RCV1 dataset.

Chapter 5: Conclusions

In this thesis, we presented a novel approach to multi-class linear classification by applying HyGAMP to both an MMSE and a MAP formulation of the multinomial logistic regression problem. However, in order to make our algorithms competitive with state-of-the-art algorithms, we had to approximate HyGAMP, which resulted in Simplified-HyGAMP (SHyGAMP). In SHyGAMP, we were able to use existing methods based on EM and SURE to tune parameters of our model, which let us avoid an expensive cross-validation tuning procedure. Lastly, we compared our algorithms to two state-of-the-art multinomial logistic regression algorithms and saw ours were very competitive; in almost every case our algorithms had a lower runtime, and in most cases our algorithms achieved nearly the same or better error rates as the competition.

Appendix A: Derivation for the Bayes' error rate

This appendix contains the derivation for the Bayes' error rate (BER) (denoted by ε_B) given in (1.27). The BER is the probability of error that is achieved with Bayes' optimal classifiers. We are interested in this quantity given the parameters of our data model, c and σ_a^2. Recall $\mathsf{z} = \boldsymbol{X}^\mathsf{T}\mathbf{a}$. Let $\boldsymbol{X} = \frac{1}{\sigma_a^2}[\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_D]$ (i.e., our weight vector $\overline{\boldsymbol{x}}_d = \frac{1}{\sigma_a^2}\boldsymbol{\mu}_d$ for all d) in accordance with (1.32)). Then, the joint distribution of $\mathsf{z} \,|\, \mathsf{y}$ is

$$\mathcal{N}(\boldsymbol{z}; \boldsymbol{X}^\mathsf{T}\boldsymbol{\mu}_y, \boldsymbol{X}^\mathsf{T}\sigma_a^2\boldsymbol{I}_D\boldsymbol{X}). \tag{A.1}$$

Due to the equal length and mutually orthogonal class mean assumption, $\mathsf{z} \,|\, \mathsf{y}$ has a mean containing all zeros except for $\frac{c^2}{\sigma_a^2}$ in the yth position and covariance matrix $\frac{c^2}{\sigma_a^2}\boldsymbol{I}_D$, which is invariant to y.

The average probability of classification error is

$$\begin{aligned}
\varepsilon_B &= \Pr(\widehat{y}(\mathbf{a}) \neq \mathsf{y}) && \text{(A.2)}\\
&= \sum_{d=1}^{D} \Pr(\widehat{y}(\mathbf{a}) \neq d \,|\, \mathsf{y} = d)\Pr(\mathsf{y} = d) && \text{(A.3)}\\
&= \frac{1}{D}\sum_{d=1}^{D} \underbrace{\Pr(\widehat{y}(\mathbf{a}) \neq d \,|\, \mathsf{y} = d)}_{\triangleq \varepsilon_B \,|\, \mathsf{y}=d} && \text{(A.4)}
\end{aligned}$$

The error probability conditioned on y can be written as

$$\varepsilon_B \,|\, (\mathsf{y} = d) = 1 - \Pr(\widehat{y}(\boldsymbol{a}) = d \,|\, \mathsf{y} = d) \tag{A.5}$$

$$= 1 - \Pr(d = \arg\max_{d'} \mathsf{z}_{d'} \,|\, \mathsf{y} = d) \tag{A.6}$$

$$= 1 - \Pr(\mathsf{z}_d > \mathsf{z}_1, \mathsf{z}_d > \mathsf{z}_2, ..., \mathsf{z}_d > \mathsf{z}_{d-1}, \mathsf{z}_d > \mathsf{z}_{d+1}, ..., \mathsf{z}_d > \mathsf{z}_D \,|\, \mathsf{y} = d) \tag{A.7}$$

The events $\{\mathsf{z}_d > \mathsf{z}_{d'}\}_{d' \neq d}$ in the joint probability expression above are not independent because they all involve the random variable z_d. We break the dependence among terms by conditioning on z_d and averaging over its distribution.

$$\varepsilon_B \,|\, (\mathsf{y} = d) =$$
$$1 - \int p_{\mathsf{z}_d}(z) \Pr(z > \mathsf{z}_1, z > \mathsf{z}_2, ..., z > \mathsf{z}_{d-1}, z > \mathsf{z}_{d+1}, ..., z > \mathsf{z}_D \,|\, \mathsf{y} = d)\, \mathrm{d}z \tag{A.8}$$

By conditioning on $\mathsf{y} = d$, the random variables $\{\mathsf{z}_{d'}\}_{d' \neq d}$ are distributed i.i.d. $\mathcal{N}(0, c^2/\sigma_a^2)$ and thus

$$\varepsilon_B \,|\, (\mathsf{y} = d) = 1 - \int \underbrace{p_{\mathsf{z}_d}(z)}_{\mathcal{N}(z; c^2/\sigma_a^2, c^2/\sigma_a^2)} \prod_{d'=1, \neq d}^{D} \underbrace{\Pr(z > \mathsf{z}_{d'} \,|\, \mathsf{y} = d)}_{\Phi\left(\frac{z}{c/\sigma_a}\right)} \mathrm{d}z \tag{A.9}$$

This simplifies to

$$\varepsilon_B \,|\, (\mathsf{y} = d) = 1 - \int \mathcal{N}(z; \frac{c}{\sigma_a}, 1) \Phi(z)^{D-1}\, \mathrm{d}z \tag{A.10}$$

Finally, since the previous expression is independent of y we have $\varepsilon_B = \varepsilon_B \,|\, (\mathsf{y} = d)$.

Appendix B: Derivations for Output Estimator Approximations in MMSE-SHyGAMP

B.1 Derivation for the Taylor series approximation

This appendix contains the derivation for (3.60), (3.61) and (3.62). Let $f(\boldsymbol{z}) \triangleq \frac{\exp(z_y)}{\sum_{d=1}^{D}\exp(z_d)}$. By approximating $f(\boldsymbol{z})$ with a second order Taylor series, we obtain

$$p_{\mathbf{z}|\mathbf{y},\mathbf{p}}(\boldsymbol{z}\,|\,y,\widehat{\boldsymbol{p}};\boldsymbol{Q}^{\mathbf{p}}) \approx C^{-1}\Bigg(f(\widehat{\boldsymbol{p}}) + \sum_{j=1}^{D}\frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j}(z_j-\widehat{p}_j) + \frac{1}{2}\sum_{j=1}^{D}\sum_{k=1}^{D}\frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_j \partial z_k}(z_j-\widehat{p}_j)(z_k-\widehat{p}_k)\Bigg)\prod_{d=1}^{D}\mathcal{N}(z_d;\widehat{p}_d,q_d^{\mathbf{p}}) \tag{B.1}$$

The terms in the gradient of $f(\cdot)$ are given by:

$$\frac{\partial f(\boldsymbol{z})}{\partial z_y} = \frac{\exp(z_y)\sum_{d=1,\neq y}^{D}\exp(z_d)}{(\sum_{d=1}^{D}\exp(z_d))^2} \tag{B.2}$$

$$\frac{\partial f(\boldsymbol{z})}{\partial z_{i\neq y}} = \frac{-\exp(z_y+z_i)}{(\sum_{d=1}^{D}\exp(z_d))^2} \tag{B.3}$$

And the terms in the Hessian that are needed in the final computations are given by:

$$\frac{\partial^2 f(\boldsymbol{z})}{\partial z_y^2} = \frac{\exp(z_y)\sum_{d=1,\neq y}^{D}\exp(z_d)(\sum_{d=1,\neq y}^{D}\exp(z_d)-\exp(z_y))}{(\sum_{d=1}^{D}\exp(z_d))^3} \tag{B.4}$$

$$\frac{\partial^2 f(\boldsymbol{z})}{\partial z_{i\neq y}^2} = \frac{-\exp(z_y+z_i)(\sum_{d=1,\neq i}^{D}\exp(z_d)-\exp(z_y))}{(\sum_{d=1}^{D}\exp(z_d))^3} \tag{B.5}$$

(It will be seen later why the off-diagonal terms of the Hessian are not needed).

In order to compute the scale constant C the following integrals must be evaluated (by distributing the product of normal distributions over the sum in (B.1)): the constant term distributed over the multivariate normal:

$$\int_{\boldsymbol{z}} f(\widehat{\boldsymbol{p}}) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.6}$$

$$= f(\widehat{\boldsymbol{p}}); \tag{B.7}$$

the linear terms:

$$\int_{\boldsymbol{z}} \sum_{j=1}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j} (z_j - \widehat{p}_j) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.8}$$

$$= \sum_{j=1}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j} \int_{\boldsymbol{z}} (z_j - \widehat{p}_j) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.9}$$

$$= \sum_{j=1}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j} \int_{z_j} (z_j - \widehat{p}_j) \mathcal{N}(z_j; \widehat{p}_j, q_j^{\mathbf{p}}) \, \mathrm{d}z_j \int_{\boldsymbol{z} \setminus z_j} \prod_{d=1, \neq j}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \setminus z_j \tag{B.10}$$

$$= 0 \text{ since the first central moment of a Gaussian is 0;} \tag{B.11}$$

and lastly the quadratic terms:

$$\int_{\boldsymbol{z}} \frac{1}{2} \sum_{j=1}^{D} \sum_{k=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_j \partial z_k} (z_j - \widehat{p}_j)(z_k - \widehat{p}_k) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.12}$$

$$= \frac{1}{2} \sum_{j=1}^{D} \sum_{k=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_j \partial z_k} \int_{\boldsymbol{z}} (z_j - \widehat{p}_j)(z_k - \widehat{p}_k) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.13}$$

$$= \frac{1}{2} \sum_{d} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_d^2} q_d^{\mathbf{p}} \tag{B.14}$$

since when $j \neq k$ the integral equals 0.

Given (B.7), (B.11), and (B.14), the normalizing constant can be approximated by:

$$C \approx f(\widehat{\boldsymbol{p}}) + \frac{1}{2} \sum_{d=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_d^2} q_d^{\mathbf{p}}. \tag{B.15}$$

Now we must compute the components of the first moment. The constant term is

$$\int_{\boldsymbol{z}} z_i f(\widehat{\boldsymbol{p}}) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.16}$$

$$= f(\widehat{\boldsymbol{p}}) \, \widehat{p}_i. \tag{B.17}$$

The linear terms are

$$\int_{\boldsymbol{z}} z_i \sum_{j=1}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j} (z_j - \widehat{p}_j) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.18}$$

$$= \int_{\boldsymbol{z}} z_i \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i} (z_i - \widehat{p}_i) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} +$$

$$\int_{\boldsymbol{z}} z_i \sum_{j=1,\neq i}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j} (z_j - \widehat{p}_j) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.19}$$

$$= \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i} \int_{\boldsymbol{z}} (z_i^2 - z_i \widehat{p}_i) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} +$$

$$\int_{\boldsymbol{z}} z_i \sum_{j=1,\neq i}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j} (z_j - \widehat{p}_j) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.20}$$

$$= \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i} (q_i^{\mathbf{p}} + \widehat{p}_i^2 - \widehat{p}_i^2) +$$

$$\sum_{j=1,\neq i}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j} \int_{z_i} z_i \mathcal{N}(z_i; \widehat{p}_i, q_i^{\mathbf{p}}) \, \mathrm{d}z_i \int_{\boldsymbol{z} \setminus z_i} (z_j - \widehat{p}_j) \prod_{d=1,\neq i}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \setminus z_i \tag{B.21}$$

$$= \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i} (q_i^{\mathbf{p}}) +$$

$$\sum_{j=1,\neq i}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j} \widehat{p}_i \int_{z_j} (z_j - \widehat{p}_j) \mathcal{N}(z_j; \widehat{p}_j, q_j^{\mathbf{p}}) \, \mathrm{d}z_j \int_{\boldsymbol{z} \setminus \{z_i, z_j\}} \prod_{d=1,\neq i,j}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \setminus \{z_i, z_j\} \tag{B.22}$$

$$= \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i} (q_i^{\mathbf{p}}). \tag{B.23}$$

And finally, the quadratic terms are

$$\int_{\boldsymbol{z}} z_i \frac{1}{2} \sum_{j=1}^{D} \sum_{k=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_j \partial z_k} (z_j - \widehat{p}_j)(z_k - \widehat{p}_k) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.24}$$

$$= \frac{1}{2} \sum_{j=1}^{D} \sum_{k=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_j \partial z_k} \int_{\boldsymbol{z}} z_i (z_j - \widehat{p}_j)(z_k - \widehat{p}_k) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.25}$$

$$= 0 \text{ when } j \neq k$$

when $j = k = i$

$$= \frac{1}{2} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_i}^2} \int_{\boldsymbol{z}} (z_i^3 - 2\widehat{p}_i z_i^2 + \widehat{p}_i^2 z_i) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.26}$$

$$= \frac{1}{2} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_i}^2} (\widehat{p}_i^3 + 3\widehat{p}_i q_i^{\mathbf{p}} - 2\widehat{p}_i(\widehat{p}_i^2 + q_i^{\mathbf{p}}) + \widehat{p}_i^3) \tag{B.27}$$

$$= \frac{1}{2} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_i}^2} (\widehat{p}_i q_i^{\mathbf{p}}) \tag{B.28}$$

else when $j = k;\ j, k \neq i$

$$= \frac{1}{2} \sum_{j \neq i} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_j}^2} \int_{\boldsymbol{z}} z_i (z_j^2 - 2z_j \widehat{p}_j + \widehat{p}_j^2) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.29}$$

$$= \frac{1}{2} \sum_{j \neq i} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_j}^2} (\widehat{p}_i(\widehat{p}_j^2 + q_j^{\mathbf{p}} - 2\widehat{p}_j^2 + \widehat{p}_j^2)) \tag{B.30}$$

$$= \frac{1}{2} \sum_{j \neq i} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_j}^2} (\widehat{p}_i q_j^{\mathbf{p}}) \tag{B.31}$$

putting the two together, the first moment of the quadratic term is

$$= \frac{1}{2} \sum_{j=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_j}^2} (\widehat{p}_i q_j^{\mathbf{p}}). \tag{B.32}$$

Given (B.17), (B.23), and (B.32), the approximate mean of the posterior is now given by

$$\widehat{z}_i \approx C^{-1} \left(f(\widehat{\boldsymbol{p}}) \, \widehat{p}_i + \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i} (q_i^{\mathbf{p}}) + \frac{1}{2} \sum_{j=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_j}^2} (\widehat{p}_i q_j^{\mathbf{p}}) \right). \tag{B.33}$$

Lastly, the components of the second moment must be computed. The constant term is

$$\int_{\boldsymbol{z}} z_i^2 f(\widehat{\boldsymbol{p}}) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.34}$$

$$= f(\widehat{\boldsymbol{p}}) \, (\widehat{p}_i^2 + q_i^{\mathbf{p}}). \tag{B.35}$$

The linear terms are

$$\int_{\boldsymbol{z}} z_i^2 \sum_{j=1}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j} (z_j - \widehat{p}_j) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.36}$$

$$= \int_{\boldsymbol{z}} z_i^2 \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i} (z_i - \widehat{p}_i) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} +$$

$$\int_{\boldsymbol{z}} z_i^2 \sum_{j=1, \neq i}^{D} \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_j} (z_j - \widehat{p}_j) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.37}$$

$$= 2 \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i} \widehat{p}_i q_i^{\mathbf{p}}. \tag{B.38}$$

And the quadratic terms are

$$\int_{\boldsymbol{z}} z_i^2 \frac{1}{2} \sum_{j=1}^{D} \sum_{k=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_j \partial z_k} (z_j - \widehat{p}_j)(z_k - \widehat{p}_k) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.39}$$

$$= \frac{1}{2} \sum_{j=1}^{D} \sum_{k=1}^{D} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_j \partial z_k} \int_{\boldsymbol{z}} z_i^2 (z_j - \widehat{p}_j)(z_k - \widehat{p}_k) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.40}$$

$$= 0 \text{ when } j \neq k$$

if $i = j = k$

$$= \frac{1}{2} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_i}^2} \int_{\boldsymbol{z}} (z_i^4 - 2 z_i^3 \widehat{p}_i + \widehat{p}_i z_i^2) \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.41}$$

$$= \frac{1}{2} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_i}^2} q_i^{\mathbf{p}} (3 q_i^{\mathbf{p}} + \widehat{p}_i^2) \tag{B.42}$$

if $i \neq j; j = k$

$$= \frac{1}{2} \sum_{j \neq i} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_j}^2} \int_{\boldsymbol{z}} z_i^2 (z_j - \widehat{p}_j)^2 \prod_{d=1}^{D} \mathcal{N}(z_d; \widehat{p}_d, q_d^{\mathbf{p}}) \, \mathrm{d}\boldsymbol{z} \tag{B.43}$$

$$= \frac{1}{2} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_j}^2} (\widehat{p}_i^2 + q_i^{\mathbf{p}}) \sum_{j \neq i} q_j^{\mathbf{p}}. \tag{B.44}$$

So, given (B.35), (B.38), and (B.44), the approximate variance is given by

$$q_i^{\mathbf{z}} \approx C^{-1} \Bigg(f(\widehat{\boldsymbol{p}}) \, (\widehat{p}_i^2 + q_i^{\mathbf{p}}) + 2 \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i} \widehat{p}_i q_i^{\mathbf{p}} + \frac{1}{2} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_i}^2} q_i^{\mathbf{p}} (3 q_i^{\mathbf{p}} + \widehat{p}_i^2) + \frac{1}{2} \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial {z_j}^2} (\widehat{p}_i^2 + q_i^{\mathbf{p}}) \sum_{j=1, \neq i}^{D} q_j^{\mathbf{p}} \Bigg) - \widehat{z}_i^2. \tag{B.45}$$

B.2 Derivation for the Gaussian posterior approximation

This appendix contains the derivation for the results in Section 3.3.2 in which we approximate the approximate posterior given in (2.4) with a Gaussian distribution.

We will rewrite the approximate posterior as

$$p_{\mathbf{z}\,|\,\mathsf{y},\mathbf{p}}(\boldsymbol{z}\,|\,y,\widehat{\boldsymbol{p}};\boldsymbol{Q}^{\mathbf{p}}) = \exp(\log p_{\mathbf{z}\,|\,\mathsf{y},\mathbf{p}}(\boldsymbol{z}\,|\,y,\widehat{\boldsymbol{p}})) \tag{B.46}$$

$$= \exp\left(\sum_{d=1}^{D}\left(-\frac{1}{2q_d^{\mathbf{p}}}(z_d-\widehat{p}_d)^2+\frac{1}{\sqrt{2\pi q_d^{\mathbf{p}}}}\right)+\log\frac{\exp(z_y)}{\sum_{d=1}^{D}\exp(z_d)}\right). \tag{B.47}$$

Constants in the exponent can be ignored because they do not affect the mean or variance. The last term in the exponent, $f(\boldsymbol{z}) \triangleq \log\frac{\exp(z_y)}{\sum_{d=1}^{D}\exp(z_d)}$, must be approximated with a quadratic. The terms in the gradient are given by:

$$\frac{\partial f(\boldsymbol{z})}{\partial z_y} = 1 - \frac{\exp(z_y)}{\sum_{d=1}^{D}\exp(z_d)} \tag{B.48}$$

$$\frac{\partial f(\boldsymbol{z})}{\partial z_{i\neq y}} = -\frac{\exp(z_i)}{\sum_{d=1}^{D}\exp(z_d)} \tag{B.49}$$

And the terms in the Hessian are given by:

$$\frac{\partial^2 f(\boldsymbol{z})}{\partial z_i^2} = -\frac{\exp(z_i)\sum_{d=1,\neq i}^{D}\exp(z_d)}{(\sum_{d=1}^{D}\exp(z_d))^2} \tag{B.50}$$

$$\frac{\partial^2 f(\boldsymbol{z})}{\partial z_i \partial z_j} = \frac{\exp(z_i+z_j)}{(\sum_{d=1}^{D}\exp(z_d))^2} \tag{B.51}$$

The approximate posterior is now further approximated by

$$p_{\mathbf{z}\,|\,\mathsf{y},\mathbf{p}}(\boldsymbol{z}\,|\,y,\widehat{\boldsymbol{p}};\boldsymbol{Q}^{\mathbf{p}}) \approx C^{-1}\exp\left(\sum_{i=1}^{D}-\frac{1}{2q_i^{\mathbf{p}}}(z_i-\widehat{p}_i)^2 + f(\widehat{\boldsymbol{p}}) + \sum_{i=1}^{D}\frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i}(z_i-\widehat{p}_i) + \frac{1}{2}\sum_{j,k=1}^{D}\frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_j \partial z_k}(z_j-\widehat{p}_j)(z_k-\widehat{p}_k)\right), \tag{B.52}$$

where C^{-1} is an arbitrary normalizing constant. Equation (B.52) can be simplified by multiplying out terms and absorbing constants into C^{-1}.

$$p_{\mathbf{z}\,|\,\mathsf{y},\mathbf{p}}(\boldsymbol{z}\,|\,y,\widehat{\boldsymbol{p}};\boldsymbol{Q}^{\mathbf{p}}) \approx C^{-1}\exp\left(\sum_{i=1}^{D}-\frac{1}{2q_i^{\mathbf{p}}}z_i^2 + \sum_{i=1}^{D}\frac{\widehat{p}_i}{q_i^{\mathbf{p}}}z_i + \sum_{i=1}^{D}\frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i}z_i + \frac{1}{2}\sum_{j,k=1}^{D}\frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_j \partial z_k}(z_j z_k - z_j\widehat{p}_k - z_k\widehat{p}_j)\right) \tag{B.53}$$

After grouping like variables the equations for $\boldsymbol{A}$ and $\boldsymbol{b}$ are

$$A_{ij} = \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_i \partial z_j} - \frac{1}{q_i^{\mathbf{p}}} \text{ if } i = j \tag{B.54}$$

$$A_{ij} = \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_i \partial z_j} \text{ if } i \neq j \tag{B.55}$$

$$b_i = \frac{\widehat{p}_i}{q_i^{\mathbf{p}}} - \sum_{j=1}^{D} \widehat{p}_j \frac{\partial^2 f(\widehat{\boldsymbol{p}})}{\partial z_i \partial z_j} + \frac{\partial f(\widehat{\boldsymbol{p}})}{\partial z_i}. \tag{B.56}$$

Bibliography

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2007.

[2] 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*, 293:2425–2430, 2001.

[3] S. Ryali, K. Supekar, D. A. Abrams, and V. Menon. Sparse logistic regression for whole-brain classification of fmri data. *NeuroImage*, 51:752–764, 2010.

[4] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, April 2004.

[5] H. Sun, A. Hui, Q. Su, A. Vortmeyer, Y. Kotliarov, S. Pastorino, A. Passaniti, J. Menon, J. Walling, R. Bailey, M. Rosenblum, T. Mikkelsen, and H. Fine. Neuronal and glioma-derived stem cell factor induces angiogenesis within the brain. *Cancer Cell*, 9:287–300, 2006.

[6] E. P. Xing, M. I. Jordan, and R. M. Karp. Feature selection for high-dimensional genomic microarray data. *Intl Wkshp. Mach. Learn.*, pages 601–608, 2001.

[7] J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):550–554, 1994. Data taken from http://www.gaussianprocess.org/gpml/data/.

[8] H. V. Poor. *An Introduction to Signal Detection and Estimation*. Springer, New York, 2nd edition, 1994.

[9] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(6):957–968, June 2005.

[10] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, 3rd edition, 1991.

[11] Sundeep Rangan. Generalized approximate message passing for estimation with random linear mixing. *arXiv:1010.5141*, October 2010.

[12] Sundeep Rangan, Alyson K. Fletcher, Vivek K Goyal, and Philip Schniter. Hybrid approximate message passing with applications to structured sparsity. *arXiv:1111.2581*, November 2011.

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

[14] A. Genkin, D. D. Lewis, and D. Madigan. Large-scale Bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, August 2007.

[15] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *J. Statst. Softw.*, 33(1):1–22, January 2010.

[16] G. C. Cawley, N. L. C. Talbot, and M. Girolami. Sparse multinomial logistic regression via Bayesian L1 regularisation. In *Proc. Neural Inform. Process. Syst. Conf.*, pages 209–216, 2007.

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

[18] Yves Grandvalet. Least absolute shrinkage is equivalent to quadratic penalization. In *Proc. Int. Conf. Artific. Neural Netw.*, pages 201–206, 1998.

[19] David J. C. MacKay. The evidence framework applied to classification networks. *Neural Comput.*, 4:720–736, 1992.

[20] Justin Ziniel, Philip Schniter, and P. Sederberg. Binary classification and feature selection via generalized approximate message passing. *IEEE Trans. Signal Process.*, 63(8):2020–2032, 2015.

[21] Robert Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. B*, 58(1):267–288, 1996.

[22] D. R. Hunter and K. Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

[23] D. Hunter and R. Li. Variable selection using MM algorithms. *Ann. Statist.*, 33(4):1617–1642, 2005.

[24] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[25] R. Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill, 3rd edition, 1999.

[26] Ali Mousavi, Arian Maleki, and Richard G. Baraniuk. Parameterless, optimal approximate message passing. *arXiv:1311.0035*, November 2013.

[27] L. A. Stefanski. A normal scale mixture representation of the logistic distribution. *Stats. Prob. Letters*, 11(1):69–70, 1991.

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

[29] J. P. Vila and P. Schniter. Expectation-maximization Gaussian-mixture approximate message passing. *IEEE Trans. Signal Process.*, 61(19):4658–4672, Oct. 2013.