Fitting the Discrete Log-normal Model

STATS 606 Project

Mason Ferlic and Easton Huch

March 2022

1 Introduction

Count data arise frequently in many scientific and industrial settings. Often, the Poisson model serves as the starting point for analyzing these data sets, due largely to its simplicity and interpretability. However, the Poisson model makes a strong statistical assumption that often proves problematic: the equivariance assumption. Under this assumption, that variance of the count data must equal the mean, which is often not the case.

In many applications, data analysts find that the variance of their collected data set is substantially larger than the mean, a situation referred to as overdispersion. Historically, overdispersion has received a great deal of attention due to (1) its ubiquity and (2) theoretical reasons why it might occur (see, for example, the discussion in [4]). Perhaps the two most common statistical models employed in this setting are the negative binomial model and the quasipoisson model.

Underdispersion, though less common, can also have undesirable implications for statistical analysis. For this reason, researchers have also developed models that are specifically designed for underdispersed count data (see [6] for a comparison of some of these models).

Statistical models that work well for both underdispersion and overdispersion are relatively less common and have some notable downsides. For example, the quasipoisson model does not involve a true likelihood function, which may be undesirable for certain use cases; e.g., generating prediction intervals. As another example, the COM-Poisson model is extremely flexible and a member of the exponential family; however, its moments are not available in closed form and must be approximated.

In this project, we explore an alternative model that Easton proposed in STATS 608 and fit using Bayesian methods. We call it the discrete log-normal model because it involves a latent log-normal random variable that is discretized in a deterministic fashion. In Section 2, we motivate and formulate the model. In Section 3, we propose two families of methods for fitting the model. In Section 4, we apply the methods to simulated data sets and compare them. We close with a discussion of the results in Section 5.

2 Statistical Model

Mathematically, the discrete log-normal distribution takes the following form:

$$
\begin{aligned}
Y_i &= \lfloor \exp(Z_i) \rfloor \\
Z_i &\sim \text{Normal}(\mu_i, \sigma_i) \\
\mu_i &= x_i'\beta \\
\sigma_i &= \exp(w_i'\alpha)
\end{aligned}
\tag{1}
$$

The model uses a vector generalized linear model [5] (VGLM) for a latent variable, Z_i. The variable Z_i is then exponentiated and rounded down to obtain a count, $Y_i \in \{0, 1, 2, ...\}$. x_i is a vector of predictors for the mean function and w_i is a vector of predictors for the variance function. While unconventional, the model offers several compelling benefits:

1. The model naturally allows for underdispersion and overdispersion by adjusting the magnitude of α
2. The model is not restricted to a fixed mean-variance relationship; instead it allows the variance to change as a flexible function of predictors
3. The model could be extended to account for correlation (e.g., due to temporal proximity) by assuming the latent Z_i's are multivariate normal with nonzero covariance.
4. Algorithms for fitting this model should be relatively straightforward because the distribution of the latent variable is Gaussian

3 Optimization Algorithms

We fit the model in Section 2 using L_2-penalized maximum likelihood estimation. We use two broad families of optimization algorithms to fit the model. The first family is to directly optimize the resulting loss function using gradient-based methods. The second is to use the expectation-maximization algorithm (see [2] for an overview of this algorithm). We discuss these families of algorithms in detail in Sections 3.1 and 3.2, repectively.

3.1 Direct Gradient-Based Methods

The gradient derivations for the first two model-fitting approaches are facilitated by writing the marginal likelihood of Y_i as follows:

$$
p\left(y_i \,|\, \alpha, \beta\right) = \int_{\log(y_i)}^{\log(y_i+1)} f(t \,|\, \mu_i, \sigma_i)\, dt = F\left\{\log(y_i + 1) \,|\, \mu_i, \sigma_i\right\} - F\left\{\log(y_i) \,|\, \mu_i, \sigma_i\right\} \tag{2}
$$

Where f and F denote the pdf and cdf of the normal distribution with parameters μ_i and σ_i. The penalized log-likelihood function is then given by

$$
\ell(\alpha, \beta) = \sum_{i=1}^{n} \log p\left(y_i \,|\, \alpha, \beta\right) - \frac{\lambda}{2}\left(\alpha'\alpha + \beta'\beta\right) \tag{3}
$$

As is typical, we omit the penalty for the intercept in β; however, we include it for the intercept in α because doing so improves the stability of our model-fitting algorithm. The gradient and Hessian of $\ell(\alpha, \beta)$ are provided in Appendix A.

3.1.1 Implementation of Model Fitting Algorithms

We fit our discrete log-normal model by extending the GenericLikelihoodModel class implemented in the StatsModel package in Python. The GenericLikelihoodModel class allows the fitting of any likelihood function via maximum likelihood optimization. Our extended class has access to a range of gradient and gradient-free solvers. Optimization methods that require only the likelihood function are Nelder-Mead and Powell. Methods that require a gradient are BFGS, conjugate gradient, and Newton-conjugate gradient. The Newton method requires both the gradient and Hessian. Having an analytical expression for the Hessian gives us standard errors of the parameter estimates based on the observed information matrix. Including a small penalty with L_2-regularization helps improves the stability of the model fitting algorithms and ensures the Hessian is positive-definite thus invertible for estimates of the standard errors.

Numerical stability is an issue when first fitting the algorithms due to catastrophic cancellation in the *logcdf* terms for very large or very small probability values. We solve this by expanding the *log* terms to help with numerical stability and keep machine precision.

To improve the convergence properties of the five tested optimization methods we seed the starting parameters for the mean coefficients with the parameter estimates from a regular Poisson model fit to the data. This is not necessary for all the solvers but is critical for the Newton method. Upon testing, the Newton-conjugate gradient method appears to be the most robust to initialization and L_2 penalty values, whereas the Newton method is very sensitive to starting parameters and values not near the true values often cause the algorithm to fail to converge.

3.2 Expectation Maximization Algorithms

Expectation maximization algorithms for our model involve calculating the penalized expected log-likelihood of Y and Z, fixing the parameters to their current working estimates. The expectation is taken with respect to the distribution of the Z given both the parameters and the Y. Because the full expectation decomposes as a sum of expectations of i, it suffices to compute the distribution for one value of i. This conditional turns out to be a truncated normal distribution.

Intuitively, the math works out in this manner due to the floor function in our model. Given that $Y_i = y_i$, we know that Z_i must fall between $\exp(y_i)$ and $\exp(y_i + 1)$. The details of this result and other technical results in this section are available in Appendix B.

Since Z_i given Y_i and the parameters is a truncated normal distribution, the expectation of interest depends on the moments of the truncated normal distribution which, conveniently, are available in closed form. Denoting the first and second moments of this distribution as e_{1i} and e_{2i}, the full expectation becomes

$$h(\alpha, \beta) = \sum_{i=1}^{n} \Big[-\log(\sigma_i) - \frac{1}{2\sigma_i^2}\big(e_{2i} - 2\, e_{1i}\, \mu_i + \mu_i^2\big)\Big] - \frac{\lambda}{2}\left(\beta'\beta + \alpha'\alpha\right) \tag{4}$$

Then next step of the algorithm is to maximize this quantity with respect to α and β. Comparing this equation to the log-likelihood function for normal linear regression, we see that the optimal value of β is given by an L^2-penalized weighted least squares regression, the solution to which is given by

$$\hat{\beta} = \left(X'\Sigma^{-1}X + \lambda I\right)^{-1} X'\Sigma^{-1}e_1 \tag{5}$$

Where Σ is a diagonal matrix with the i-th diagonal element equal to σ_i^2. e_1 is a column vector with i-th element e_{1i}.

In contrast, the optimal value of α does not have a simple closed-form solution. Instead, we must use gradient-based optimization techniques such as those applied in 3.1. We tested two optimization algorithms for this problem: a first-order version using gradient ascent with back-tracking line search and a second-order version using Newton's method. The details of both of these methods are available in Appendix B.

4 Results

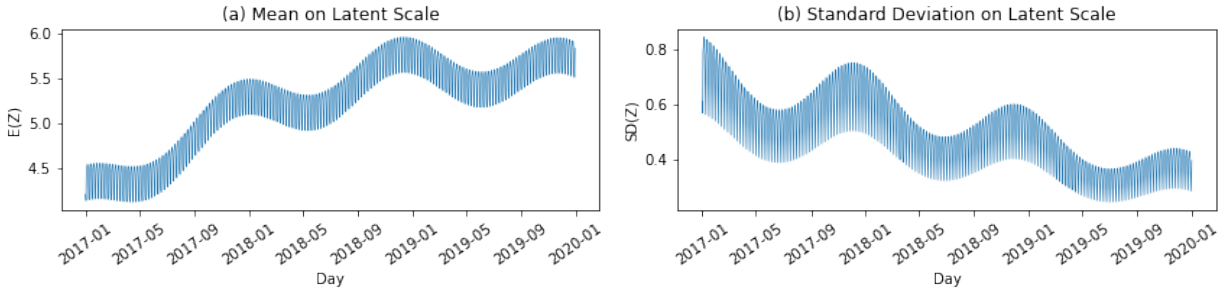
In this section, we simulate data from our model and estimate its parameters using the techniques discussed in Section 3. The model structure for our simulation is based on time series count data, which arise commonly in online web-tracking. A typical objective in this setting is to generate short-horizon (e.g., one-week) predictions and prediction intervals in order to identify 'anomalies' in incoming data.

In our simulation, the predictors for the mean function are the same as the predictors for the variance function; i.e., $x_i = w_i$. Each covariate vector is composed of the following terms:

- Intercept
- Linear time trend (centered and scaled)
- Quadratic time trend (centered and scaled)
- Six day-of-the-week indicators (Sunday is the omitted group)
- Two trig functions (sine and cosine) meant to simulate seasonality; the input to these functions is scaled so that one year corresponds with one period

In total, the model include 22 parameters. Their values were selected such that (1) the resulting time series are similar to those observed in practice and (2) the mean-variance relationship is more sophisticated than typical count models allow. Note in Figure 1 that, although the mean of the latent variable Z generally increases over time, its standard deviation decreases.

Figure 1: The mean (a) and standard deviation (b) of our latent variable Z under the true parameters in our simulation. Although the mean increases over time, the standard deviation decreases, creating a complex mean-variance relationship.



In Section 4.1, we simulate one data set from our model and fit two models: (1) the true discrete log-normal model and (2) a competitor model named the GP-1. In Section 4.2, we simulate 100 data sets and fit each using the seven optimization approaches detailed in Section 3. We then compare the algorithms in terms of their runtime and the quality of their returned solution. In both sections, we impose a small penalty of only 10^{-4}.

4.1 Model Comparison Case Study

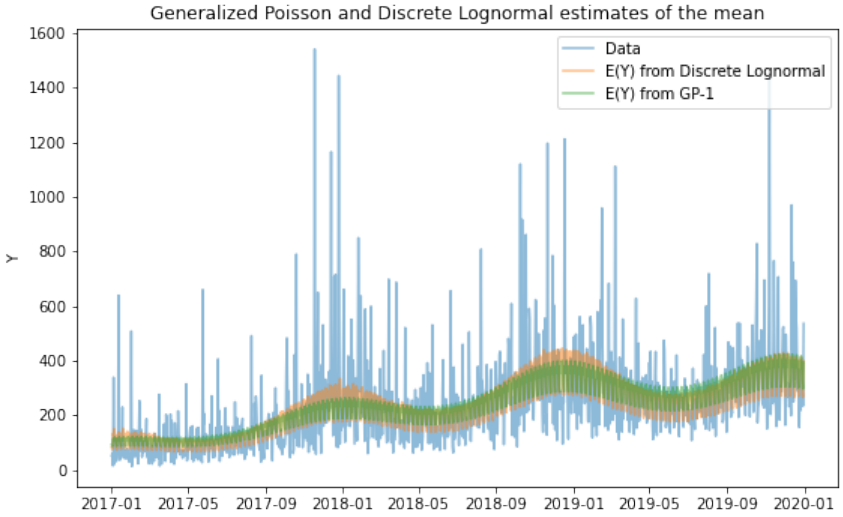
Here we compare our discrete log-normal model (DLN) to three other models commonly used to fit count data. The Poisson model follows a standard Poisson distribution but has the disadvantage that the mean is equal to the variance–a strict assumption in most real world datasets where there is over or under-dispersion. The generalized Poisson and negative binomial models attempt to correct this issue by reparameterizing the variance function to be decoupled from the mean. We test the ability of each model to handle data with both over and under-dispersion by simulating 100 realizations of time-series count data and fit each model. We then predict the mean estimates on a new dataset and compute the mean-squared prediction error in Table 1.

Discrete Model	DLN	POI	GP1	NB1
MSPE	20628.715443	20677.349160	21258.058928	20892.076206
std	3021.524422	3030.379111	3157.533030	3115.621102

Table 1: Table showing the mean squared prediction error for four different models fit to the data: discrete log-normal (DLN), poisson (POI), generalized poisson (GP1), and negative binomial (NB). Our DLN model fits the data best.

Our DLN model performs better than the standard discrete models as we are able to model a variance that is not simply a function of the mean. Figure 2 shows the predicted mean of our model and the generalized Poisson. Notice for the GP model the coverage probability scales with the means whereas our model can adjust for areas of over and under-dispersion. Figure 3 shows the predicted variance in the outcome for out DLN and the GP model. Clearly, the GP parameterization cannot accurately model data with such characteristics.

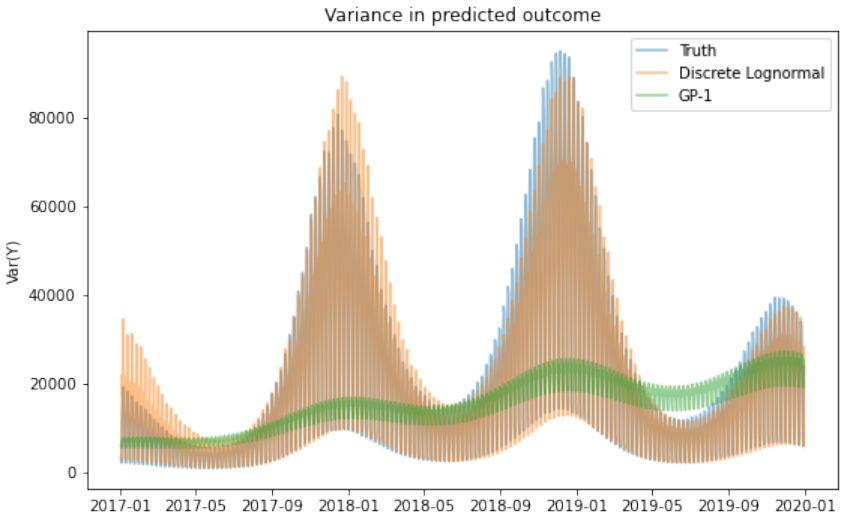
Figure 2: Figure showing the estimated mean outcome for our DLN and the generalized Poisson model fit to the same dataset.



4.2 Algorithm Comparison Simulation

In this section, we generate 100 data sets from our model, each with the same set of true parameters. We then fit the discrete log-normal model using the seven optimization algorithms given in Section 3. We ran each algorithm until the penalized log-likelihood (given in Equation (3)) improved by less than 10^{-8}.

Figure 3: Figure showing the variance in the predicted mean for the discrete log-normal and generalized Poisson models. The DLN is able to capture complex count dispersion whereas the GP model coverage probability scales with the mean.



Algorithm	EM2	bfgs	newton	ncg	cg	powell	EM1
Mean Runtime (sec)	0.042	0.098	0.126	0.192	0.392	0.464	0.703

Table 2: Table showing the mean runtimes for the seven algorithms compared in our simulation study, sorted by the mean runtime.

The results are summarized in Figure 4 and Table 2. Plot (a) in figure 4 visually compares the runtimes for the different algorithms, and the table shows the mean runtime for each algorithm. These results show that the fastest algorithm is the second-order EM algorithm, which takes less than half as much time as the next two fastest algorithms: BFGS and the Newton method. The first-order EM algorithm is the slowest algorithm, probably because the back-tracking line search is a time-consuming process.

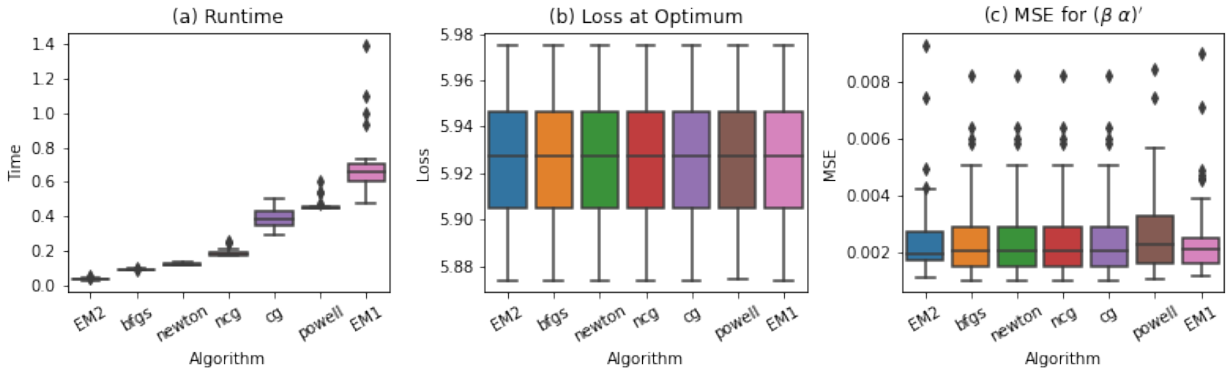
Plots (b) and (c) in the figure compare the algorithms in terms of the quality of their returned solutions. Plot (b) shows the negative penalized log-likelihood (labeled as "Loss" for brevity). The fact that these boxplots are almost visually indistinguishable provides evidence that all seven algorithms are converging to equally good solutions. Plot (c) shows the MSE in estimating α and β. We see more variability in this plot than in plot (b); however, the differences are likely negligible, especially in view of the similar loss values.

5 Discussion

In this project, we explored a new and extremely flexible statistical model for count data, and we derived two general families of methods for fitting the model via penalized maximum likelihood estimation: (1) direct gradient-based methods and (2) EM algorithms. In our simulations, we found that both approaches perform well but that the second-order EM algorithm is the most computationally efficient (at least for the size of data we were using). Second-order methods often scale poorly with data size, especially in high dimensions, so the first-order methods may be preferable with larger data sets.

One exciting avenue for future research would be to extend the discrete log-normal model to allow for correlation between the observations. This change could be accommodated by introducing correlation into

Figure 4: Comparison of the algorithms given in Section 3 on 100 simulated data sets. Plot (a) shows a great deal of variation in run time among the algorithms with the second-order EM algorithm (EM2) performing the best. On the other hand, plots (b) and (c) provide evidence that the algorithms converge to nearly identical solutions because they achieve similar losses (a) and MSE in estimating the parameters (b).



the latent Gaussian variable Z. Adapting the EM algorithm to account for this change would be particularly straightforward. The E-step would require calculating expectations with respect to a truncated multivariate Gaussian distribution (for which there are readily available algorithms and software [3]). The update for β would remain essentially unchanged (though Σ would no longer be diagonal). The gradient and Hessian for α would require a bit more effort, but should also be tractable.

References

[1] Samuel Kotz, Norman L Johnson, and N Balakrishnan. *Continuous Univariate Distributions, Vol. 1 of Wiley Series in Probability and Statistics*. 1994.

[2] Todd K Moon. "The expectation-maximization algorithm". In: *IEEE Signal processing magazine* 13.6 (1996), pp. 47–60.

[3] Stefan Wilhelm and BG Manjunath. "tmvtnorm: A package for the truncated multivariate normal distribution". In: *sigma* 2.2 (2010), pp. 1–25.

[4] Joseph M Hilbe. *Negative binomial regression*. Cambridge University Press, 2011.

[5] Thomas W Yee. *Vector generalized linear and additive models: with an implementation in R*. Vol. 10. Springer, 2015.

[6] Kimberly F Sellers and Darcy S Morris. "Underdispersion models: Models that are 'under the radar'". In: *Communications in Statistics-Theory and Methods* 46.24 (2017), pp. 12075–12086.

A Gradient and Hessian Calculations

We drop the i subscript in this section for simplicity. We also define the following quantities for ease of notation:

- $\underline{z} = (\log(y) - \mu)/\sigma$
- $\bar{z} = (\log(y+1) - \mu)/\sigma$
- $\Phi(\cdot)$: Standard normal cdf
- $\phi(\cdot)$: Standard normal pdf
- $\kappa_q = \dfrac{\bar{z}^q\,\phi(\bar{z}) - \underline{z}^q\,\phi(\underline{z})}{\Phi(\bar{z}) - \Phi(\underline{z})}$, for $q \in \{0, 1, 2, 3\}$

Working from equation (2), the gradient of $\log p\,(y_i \,|\, \alpha, \beta)$ with respect to β is

$$\nabla_\beta \log p\,(y \,|\, \alpha, \beta) = -\frac{\kappa_0}{\sigma} x \tag{6}$$

Similarly, the gradient with respect to α is

$$\nabla_\alpha \log p\,(y \,|\, \alpha, \beta) = -\kappa_1 w \tag{7}$$

Using these derivations, it's straight forward to calculate the gradient with respect to the penalized log-likelihood $\ell(\alpha, \beta)$ is:

$$\begin{aligned} \nabla_\beta\, \ell(\alpha, \beta) &= -\sum_{i=1}^{n} \frac{\kappa_{0i}}{\sigma_i} x_i - \lambda\,\beta \\ \nabla_\alpha\, \ell(\alpha, \beta) &= -\sum_{i=1}^{n} \kappa_{1i} w_i - \lambda\,\alpha \end{aligned} \tag{8}$$

Similar calculations yield the Hessian:

$$\frac{\partial}{\partial\beta\partial\beta'}\, \ell(\alpha, \beta) = -\sum_{i=1}^{n} \frac{\kappa_{0i}^2 + \kappa_{1i}}{\sigma_i^2} x_i x_i' - \lambda\, I \tag{9}$$

$$\frac{\partial}{\partial\beta\partial\alpha'}\, \ell(\alpha, \beta) = -\sum_{i=1}^{n} \frac{\kappa_{2i} + \kappa_{0i}\,(\kappa_{1i} - 1)}{\sigma_i}\, x_i w_i' \tag{10}$$

$$\frac{\partial}{\partial\alpha\partial\alpha'}\, \ell(\alpha, \beta) = -\sum_{i=1}^{n} [\kappa_{1i}\,(\kappa_{1i} - 1) + \kappa_{3i}]\, w_i w_i' - \lambda\, I \tag{11}$$

One potential challenge in performing these calculations is that the numerator and denominator in κ_q are often quite small, so a naive implementation can easily result in divide-by-zero errors as the denominator

could, without too much difficulty, be computationally zero. One way of circumventing this issue is to perform the calculations on the log scale. Doing so requires carefully using the following identity for both numerator and denominator: Given $0 < a < b$,

$$\log(b - a) = \log\left[a(b/a - 1)\right] = \log(a) - \log(b/a - 1) \tag{12}$$

The ratio b/a can also be written as $\exp\{\log(b) - \log(a)\}$. Taken together, these identities allow one to compute κ_q using the log-PDF and log-CDF of the normal distribution, both of which are readily available in common statistical computing environments. These tricks do not avoid all computational challenges associated with calculating κ_q, but we found that they noticeably reduce the potential for errors.

B Expectation-Maximization Details

In this section, we present the details of our L_2-regularized EM algorithm. The E-step in our algorithm involves calculating the following expectation:

$$h(\alpha, \beta) = \sum_{i=1}^{n} \mathbb{E}_{Z_i \sim q} \log p(Z_i, y_i | \theta) - \frac{\lambda}{2}\big(\alpha'\alpha + \beta'\beta\big) \tag{13}$$

Where $q = p(Z_i | \alpha, \beta, Y_i)$. We can work out this distribution as follows:

$$\begin{aligned} & p(z_i \,|\, \alpha, \beta, y_i) \\ \propto\; & p(z_i \,|\, \alpha, \beta)\, p(y_i \,|\, z_i) \\ =\; & \text{Normal}(\mu_i, \sigma_i^2)\, \text{I}\{y_i = \lfloor \exp(z_i) \rfloor\} \end{aligned} \tag{14}$$

Thus, given $Y_i = y_i$, Z_i follows a truncated normal distribution with parameters μ_i, σ_i^2, $\log(y_i)$, and $\log(y_i{+}1)$, where the latter two parameters indicate the interval of truncation. Calculating the above expectation requires the first and second moments of a truncated normal random variable which can be found in section 10.1 of [1]. They involve the ratios κ_0 and κ_1 defined in the previous appendix. For brevity, we denote these first and second moments (which depend on parameter estimates from the previous iteration of the EM algorithm) as e_{1i} and e_{2i}, respectively. Equation (4) then follows from this result. The optimal value for β given in Equation (5) can then be found by taking derivatives and setting them to zero.

We now provide the gradient and Hessian of $h(\alpha, \beta)$ with respect to α:

$$\begin{aligned} \nabla_\alpha h(\alpha, \beta) &= W'(\Sigma^{-1}c - \mathbb{1}) - \alpha \\ \frac{\partial}{\partial \alpha \partial \alpha'}\, h(\alpha, \beta) &= -2W'\Sigma^{-1}CW - \lambda I \end{aligned} \tag{15}$$

Where C is a diagonal matrix with i-th element equal to $e_{2i} - 2\, e_{1i}\, \mu_i + \mu_i^2$, $c = \text{diag}(C)$, and $\mathbb{1}$ is a vector of ones. As mentioned in the main text, we use these results to run two versions of the EM algorithm: a first-order version using gradient ascent with back-tracking line search and a second-order version using Newton's method. With each method, we update α only once per iteration of the EM algorithm.

In our gradient ascent algorithm, we employed a step-size of 0.001. In the back-tracking line search, we cut the step size in half until achieving our sufficient ascent condition, which required that $h(\alpha, \beta)$ improve by at least half of the amount predicted by a first order Taylor series; i.e., until

$$h\left(\hat{\alpha}^{(t)} + \eta\, d^{(t)},\ \hat{\beta}^{(t)}\right) - h\left(\hat{\alpha}^{(t)},\ \hat{\beta}^{(t)}\right) \geq \frac{\eta}{2} ||d^{(t)}||_2^2 \tag{16}$$

Where $\hat{\alpha}^{(t)}$ and $\hat{\beta}^{(t)}$ denote the estimates of α and β at iteration t in the EM algorithm, η is the step size, and $d = \nabla_\alpha h(\alpha, \beta)|_{\alpha = \hat{\alpha}^{(t)}, \beta = \hat{\beta}^{(t)}}$.

The Newton algorithm was much more straightforward. In each iteration of the EM algorithm, we set $\hat{\alpha}^{(t+1)}$ as follows:

$$\hat{\alpha}^{(t+1)} = \hat{\alpha}^{(t)} + \left(Q^{(t)}\right)^{-1} d^{(t)} \tag{17}$$

Where $Q^{(t)} = \frac{\partial}{\partial \alpha \partial \alpha'} h(\alpha, \beta)|_{\alpha=\hat{\alpha}^{(t)}, \beta=\hat{\beta}^{(t)}}$, the form of which is given in equation (15).

With both of these methods, we opted to update α only once per iteration of the EM algorithm. Doing so resulted in satisfactory convergence and, at least for the second-order method, competitive runtimes compared to the direct gradient-based methods.