Finite Mixture Models: Beyond the EM Algorithm

Carlos Gómez-Uribe
cgu@stanford.edu. ID: 005099017
Viktor Krapivin
krapivin@stanford.edu ID: 006264803
Grace Ann Woods
grawoods@stanford.edu ID:: 006279874

1 Introduction and Related Work

Finite mixture models (FMMs), such as a mixture of Gaussians, have been extensively used and studied for several decades. More recently, a new class of models called variational autoencoders (VAEs) have shown significant success as generative models of real-world data such as images [1]. VAEs can be seen as an infinite-dimensional extension of FMMs, i.e., the limit of an FMM when the number of mixture components goes to infinity. FMMs and VAEs share a non-convex likelihood function that is often thought to be intractable. To learn these models, the intractable likelihood is replaced by a lower bound that is easier to optimize,e.g., resulting in the EM algorithm (introduced in [2]) for FMMs and in variational inference (VI) for VAEs. One of the authors derived, but has not implemented, a new non-variational algorithm for VAEs that does not work with a lower bound of the likelihood (C.G.U.), but rather with an approximation of it based on the delta method. The goal of this project is to develop and evaluate a similar new method to learn FMMs, since they are simpler than VAEs, to then determine whether to continue to pursue related ideas for VAEs.

The EM algorithm for FMMs is guaranteed to be monotonically non-decreasing in the true objective, and will converge linearly to a local optimum of the log-likelihood; see [5] for a good reference on the EM and related variants. We obtain several new learning algorithms for FMMs based on a delta-method approximation of the likelihood. The gradients of the resulting objective are simple expressions that are easy to evaluate, and enable first-order learning methods. But to our surprise, we also find that the supposed-intractability of the likelihood is not so. Indeed, we obtain simple expressions for the gradient of the log-likelihood with respect to the parameters that enable any first-order learning algorithm, e.g., simple stochastic gradient descent. We also explored the possibility of second-order methods such as Newton or approximate Bayesian inference, both on the log-likelihood as well as on its new approximation. Due to space considerations we do not include them in this writeup.

2 Finite Mixture Models

Consider a mixture model with k mixture components, each following an exponential family distribution with sufficient statistic $T(x) \in \mathbb{R}^d$. The generative process is:

$$z^{(i)} \sim \text{Categorical}(\phi) \tag{1}$$

$$x^{(i)}|z^{(i)} = j \sim \text{Exp}(\eta_j), \tag{2}$$

where $\phi \in \mathbb{R}^k$ has entries $\phi_j = P(z^{(i)} = j)$ that depend on the parameters $\theta \in \mathbb{R}^{k-1}$ through

$$\phi_j = \frac{e^{\theta_j}}{1 + \sum_{i=1}^{k-1} e^{\theta_j}}, \text{ for } j = 1, \ldots, k-1\,, \quad \phi_k = \frac{1}{1 + \sum_{i=1}^{k-1} e^{\theta_j}}. \tag{3}$$

We parametrize the distribution for $z^{(i)}$ using θ rather than ϕ, because the latter take values in the simplex, while the former can take any value on $\mathbb{R}^{k-1}$. Equation 2 means that x conditioned on $z = j$ follows an exponential family distribution with natural parameter $\eta_j \in \mathbb{R}^d$, i.e., that

$$\log P\big(x^{(i)}|z^{(i)} = j\big) = \eta_j' \Phi_j^{-1} T(x^{(i)}) - b_j\big(\eta_j, \Phi_j\big) + c_j\big(x^{(i)}, \Phi_j\big), \tag{4}$$

where $'$ denotes vector or matrix transpose, $T(x^{(i)})$ denotes the sufficient statistics of $x^{(i)}$, and the known nuissance parameters Φ_j are appropriately-sized square matrices (e.g., the covariance for a Gaussian of known covariance but unknown mean). The scalar functions b_j determine the mean and variance of $T(x^{(i)})$ given $z^{(i)} = j$, through

$$h_j = E[T(x)|\eta_j] = \Phi \frac{\partial b_j(\eta_j, \Phi_j)'}{\partial \eta_j}, \tag{5}$$

$$\Sigma_j = Var(T(x)|\eta_j) = \Phi_j \frac{\partial^2 b_j(\eta_j, \Phi_j)}{\partial \eta^2} \Phi_j. \tag{6}$$

Given a choice of exponential family member, the functions $T(), b_j(), c_j()$, as well as the nuisance parameter, Φ are all known. The FMM model parameters are then $\gamma = \{\theta, \eta_1, ..., \eta_k\}$, and our goal is to learn them from independent samples of $x^{(i)}$ generated as described above. As a running example throughout, we consider a mixture of Gaussians model, where $x^{(i)}|z^{(i)} = j \sim \mathcal{N}\big(\mu_j, \Sigma_j\big)$.

2.1 Notation

Next, we introduce some notation for the main quantities we work with. Consider a single data sample x.

- Let $p_j(x) = P(x|z = j)$, i.e., $p_j(x)$ is the (conditional) density of x when generated from mixture component j.
- Let $p(x) = \sum_{i=1}^{k} \phi_i p_i(x)$ be the (unconditional) density of x.
- Let $h = E[T(x)]$ and $h_j = E[T(x)|z = j]$ be the unconditional and conditional mean of the sufficient statistic.
- Let $e(x) = T(x) - h$, and $e_j(x) = T(x) - h_j$ be the unconditional and conditional errors. Note that $E[e(x)] = 0$ and $E[e_j(x)|z = j] = 0$.
- Let $\Sigma = E[e(x)e'(x)]$ and $\Sigma_j = E[e_j(x)e'_j(x)|z = j]$ denote the unconditional and conditional covariance of $T(x)$.
- Let $\alpha_j(x) = \frac{p_j(x)}{p(x)}$. Note that $E[\alpha_j(x)] = 1$, $E[\log(\alpha_j(x))] = -KL\big(p(x), p_j(x)\big)$, and $E[\log(\alpha_j(x))|z = j] = KL\big(p_j(x), p(x)\big)$, where $KL\big(p(x), q(x)\big)$ is the KL divergence between $p(x)$ and $q(x)$. Let $\alpha(x) = [\alpha_1(x), ..., \alpha_k(x)]'$.
- Let $l_j(x) = \log p_j(x)$ denote the conditional log-likelihood, and $l(x) = [l_1(x), ..., l_k(x)]'$.
- Let $\bar{l}(x) = \sum_{i=1}^{k} \phi_i l_i(x)$ be the average conditional log-likelihood, $\Delta_j(x) = l_j(x) - \bar{l}(x)$, and $\Delta(x) = [\Delta_1(x), ..., \Delta_k(x)]'$. Similarly, let $\sigma_l^2(x) = \sum_{i=1}^{k} \phi_i \Delta_i(x)^2$ be the variance of the conditional log-likelihoods.

3 Learning Objectives

Consider the problem of learning the model parameters that maximize the log-likelihood of the data. For a single data point x, the log-likelihood $\ell(x)$ is given by

$$\ell(x) = \log P(x) = \log \sum_{i=1}^{k} \phi_i p_i(x). \tag{7}$$

This quantity is often referred to as the marginal log-likelihood in FMMs, and is the true objective we want to optimize. It is often assumed that computing gradients for Eq. 7 is intractable or too computational; so a common approach is to replace 7 with an objective that is easier to optimize. It is however, true, that Eq. 7 is harder to optimize than its fully observed analog, i.e., when z and x are observed.

3.1 The Evidence Lower Bound And EM

The EM algorithm approximates $\ell(x)$ with the evidence lower bound (ELBO)

$$\text{ELBO}(x) = \sum_{j=1}^{k} q_j(x) \log \phi_j p_j(x), \tag{8}$$

where $q_j(x)$ is a variational distribution that is optimized to make the ELBO as close as possible to the true log-likelihood. Keeping $q_j(x)$ fixed, the ELBO is just the weighted log-likelihood of the fully-observed data z and x, and for the exponential family is easy to optimize. The EM iterates optimizing the ELBO over $q_j(x)$ (***E-step***), and over the model parameters γ (***M-step***). The E-step for FMMs has the simple closed form $q_j(x) = P(z = j|x) = \phi_j \alpha_j(x)$, while the M-step involves matching the weighted observed sufficient statistics to the expected ones. Note that the M-step is equivalent to finding the η_j such that the weighted sum of errors $\sum_{i=1}^{n} q_j(x^{(i)}) e_j(x^{(i)})$ is zero. The EM algorithm for FMMs is to repeat the following two steps until convergence: (E-step) For each $i = 1, \ldots, n$ and each $j = 1, \ldots, k$, set

$$q_j(x^{(i)}) = \phi_j \alpha_j(x^{(i)})$$

(M-step) For each $j = 1, \ldots, k$ find η_j such that

$$\sum_{i=1}^{n} q_j(x^{(i)}) T(x^{(i)}) = \sum_{i=1}^{n} q_j(x^{(i)}) h_j(x^{(i)})$$

3.2 A Delta Method Approximation of $\ell(x)$

Here, we consider what we think is a new approximation of the marginal log-likelihood $\ell(x)$ based on the delta method. Specifically, since $P(x) = \sum_{j=1}^{k} P(z = j, x) = E_z[P(x|z)]$, we write $\ell(x) = \log E_z[P(x|z)]$. We then approximate $P(x|z)$ with respect to z through a second-order Taylor expansion about the mean $E[z] = \phi$ and then take the expectation, resulting in

$$\ell(x) \approx \underbrace{\log P\big(x|z = \phi\big)}_{\ell_0(x)} + \underbrace{\log\left(1 + \frac{\sum_{i,j=1}^{k} \sigma_{ij}^2 \partial_{z_i z_j}^2 P\big(x|z = \phi\big)}{2P\big(x|z = \phi\big)}\right)}_{\ell_2(x)}. \tag{9}$$

Here $z_i = 1$ when $z = i$, and $z_i = 0$ otherwise, $\partial_{z_i z_j}^2$ (or sometimes $\frac{\partial^2}{\partial_{z_i z_j}}$, for clarity depending on context) denote the derivative with respect to z_i and z_j, and σ_{ij}^2 is the covariance between z_i and z_j, given ϕ. Some algebra shows that $\sigma_{ij} = \delta_{ij}\phi_i - \phi_i\phi_j$, and δ_{ij} is the Kronecker delta, equal to zero except when $i = j$, when it is equal to 1.

In this approximation, the log-likelihood is a sum of two terms, the first of which depends only on the mean of z and the second also on its variance. To the extent that $P(x|z, \beta)$ is roughly a quadratic function of z in the region of space where most of the probability mass for z lies, we expect this approximation to be accurate. In addition, if $P(x|z, \beta)$ is in fact roughly linear in z in the same region

of space, then $\ell_0(x) \gg \ell_2(x)$, so $\log P(x|z) \approx \ell_0(x)$. We refer to $\ell_0(x)$ and $\ell_2(x)$, respectively, as the zero-th and second-order terms of our log-likelihood approximation, matching the order of the derivatives with respect to z that gave rise to them. Note that there is no $\ell_1(x)$ term, since its expectation is zero, as is typical of the delta method.

We obtain expressions for $\ell_0(x)$ and $\ell_2(x)$ starting from the observation that $P(x|z) = \prod_{j=1}^{k} p_j(x)^{z_j}$, so $P(x|z = \phi) = \prod_{j=1}^{k} p_j(x)^{\phi_j}$. Some algebra then shows that

$$\ell_0(x) = \sum_{j=1}^{k} \phi_j l_j(x) = \bar{l}(x), \tag{10}$$

$$\ell_2(x) = \log\big(1 + \frac{1}{2}\sigma_l^2(x)\big). \tag{11}$$

The approximate Delta-method objective is then

$$\tilde{\ell}(x) = \bar{l}(x) + \log\big(1 + \frac{1}{2}\sigma_l^2(x)\big). \tag{12}$$

Unlike the ELBO, $\tilde{\ell}(x)$ is not guaranteed to lower bound the true objective. But we hope it is close enough to be useful, and in particular, that local maxima of $\tilde{\ell}(x)$ with respect to the parameters, are close to local maxima of $\ell(x)$.

4 Gradients And First-Order Learning Algorithms

Gradient climbing is probably the most common optimization technique used for parameter learning. For example, in stochastic gradient descent (SGD), after a suitable initialization of the parameter estimates, one samples a random data point at every iteration, computes the objective gradient with respect to the current parameter estimates, and then updates the parameters in a direction parallel to this gradient. Other gradient learning variants, such as momentum-based, use second-order dynamics in the parameter estimates, or, in the case of Langevin dynamics, add some Gaussian noise to the parameter updates. To enable any of these algorithms, all that is needed is the gradient of the objective with respect to the parameters. We pursue these gradients for $\ell(x)$ and $\tilde{\ell}(x)$ next. Using the notation already introduced, all of these can be obtained through straightforward algebra.

4.1 True Objective or ELBO

$$\frac{\partial \ell(x)}{\partial \theta_j} = \phi_j(\alpha_j(x) - 1), \quad \frac{\partial \ell(x)}{\partial \eta_j} = \phi_j \alpha_j(x) \Phi_j^{-1} e_j(x). \tag{13}$$

We found the ELBO gradients are identical to the true objective gradients. This is because in FMMs, the E-Step can be performed exactly, and makes the ELBO tight, i.e., ELBO$(x) = \ell(x)$ when $q_j(x) = \phi_j \alpha_j(x)$. It is worth noting that to obtain the ELBO gradients, one needs to compute the gradients of Equation 8 while holding $q_j(x)$ fixed, and only after substitute $q_j(x) = \phi_j \alpha_j(x)$ in the resulting expressions, and simplifying. Reversing the order of these operations does not make sense, and yields incorrect gradients.

4.2 Delta Method

We consider the gradients using only $\ell_0(x)$, as well as the higher-order approximation $\tilde{\ell}(x)$.

$$\frac{\partial \ell_0(x)}{\partial \theta_j} = \phi_j \Delta_j(x), \quad \frac{\partial \ell_0(x)}{\partial \eta_j} = \phi_j \Phi_j^{-1} e_j(x). \tag{14}$$

The above gradients suggest that using only $\ell_0(x)$ to approximate the true objective will not result in reasonable learning. In particular, Equation 14 would result in updates to each cluster that give the same weight to every data point, hampering specialization for different mixture components. But using $\tilde{\ell}(x)$ does show some promise:

$$\frac{\partial \tilde{\ell}(x)}{\partial \theta_j} = \phi_j \left(\Delta_j(x) + \frac{\Delta_j^2(x) - \sigma_l^2(x)}{2 + \sigma_l^2(x)} \right), \quad \frac{\partial \tilde{\ell}(x)}{\partial \eta_j} = \phi_j \left(1 + \frac{2\Delta_j(x)}{2 + \sigma_l^2(x)} \right) \Phi_j^{-1} e_j(x). \tag{15}$$

Using these gradients, we can obtain, e.g., a simple SGD algorithm for each objective.

4.3 Stochastic Gradient Descent

For concreteness, we use the gradients obtained above to write down the simple SGD algorithms based on the various objectives using a single learning rate of λ. The equations below specify the update in every iteration after a randomly sampled datapoint x has been selected.

True objective or ELBO

$$\theta_j^{(t+1)} = \theta_j^{(t)} + \lambda \phi_j^{(t)} (\alpha_j(x) - 1), \quad \eta_j^{(t+1)} = \eta_j^{(t)} + \lambda \phi_j^{(t)} \alpha_j(x) \Phi_j^{-1} e_j(x). \tag{16}$$

Using $\ell(x) \approx \ell_0(x)$

$$\theta_j^{(t+1)} = \theta_j^{(t)} + \lambda \phi_j^{(t)} \Delta_j(x), \quad \eta_j^{(t+1)} = \eta_j^{(t)} + \lambda \phi_j^{(t)} \Phi_j^{-1} e_j(x). \tag{17}$$

Using $\tilde{\ell}(x)$:

$$\theta_j^{(t+1)} = \theta_j^{(t)} + \lambda\phi_j^{(t)}\left(\Delta_j(x) + \frac{\Delta_j^2(x) - \sigma_l^2(x)}{2 + \sigma_l^2(x)}\right), \quad \eta_j^{(t+1)} = \eta_j^{(t)} + \lambda\phi_j^{(t)}\left(1 + \frac{2\Delta_j(x)}{2 + \sigma_l^2(x)}\right)\Phi_j^{-1}e_j(x). \tag{18}$$

In the algorithms above, $e_j(x)$, $\alpha_j(x)$, $l_j(x)$, $\Delta_j(x)$, and $\sigma_l^2(x)$ are all evaluated using the parameter values at time t. Staring at these algorithms, it seems like optimizing $\tilde{\ell}(x)$ is essentially like optimzing $\ell(x)$, but with $l_j(x)$ playing the role of $\alpha_j(x)$. So we also want to try these variants below, which are simpler computationally, and may behave similarly. Below, in the first algorithm (19) when $\Delta_j(x)$ is negative, this algorithm actually grows the error so we also aim to try the second version 20 which is better behaved.

$$\theta_j^{(t+1)} = \theta_j^{(t)} + \lambda\phi_j^{(t)}\Delta_j(x), \quad \eta_j^{(t+1)} = \eta_j^{(t)} + \lambda\phi_j^{(t)}\Delta_j(x)\Phi_j^{-1}e_j(x). \tag{19}$$

$$\theta_j^{(t+1)} = \theta_j^{(t)} + \lambda\phi_j^{(t)}\Delta_j(x), \quad \eta_j^{(t+1)} = \eta_j^{(t)} + \lambda\phi_j^{(t)}l_j(x)\Phi_j^{-1}e_j(x). \tag{20}$$

5 Mixture of Gaussians

Here, we specialize the model and algorithms to a mixture of Gaussians. First, specialise SGD algorithms (and consequentially the EM algorithm) to mixture of Gaussians. I.e., we let $x^{(i)}|z^{(i)} = j \sim \mathcal{N}\big(\mu_j, \Sigma_j\big)$.

5.1 SGD Method for Mixture of Gaussians

It is well known that a multivariate Gaussian can be expressed in exponential family form with $\Phi = I$, and

$$T(x) = \begin{bmatrix} x \\ \text{vec}(xx') \end{bmatrix}, \quad \eta = \begin{bmatrix} \eta_\mu \\ \text{vec}(\eta_\sigma) \end{bmatrix}, \quad \eta_\mu = \Sigma^{-1}\mu, \quad \eta_\Sigma = -\frac{1}{2}\Sigma^{-1}, \tag{21}$$

where vec() denotes the column-wise concatenation of columns in the matrix argument from left to right. So $\mu = -\frac{1}{2}\eta_\sigma^{-1}\eta_\mu$ and $\Sigma = -\frac{1}{2}\eta_\sigma^{-1}$. Similarly, then,

$$E[T(x)|\eta] = \begin{bmatrix} \mu \\ \text{vec}(\Sigma + \mu\mu') \end{bmatrix}, \quad e(x) = T(x) - E[T(x)|\eta] = \begin{bmatrix} x - \mu \\ \text{vec}\Big(xx' - \Sigma - \mu\mu'\Big) \end{bmatrix}. \tag{22}$$

In our model, each mixture has its natural parameters η_j. All our SGD algorithms in Section 4.3 are written in terms of the natural parameter, and substituting the error $e(x)$ above yields the various SGD algorithms in terms of the natural parameters. We can run the algorithms that way, and after convergence translate the resulting natural parameters into μ's and Σ's. We also note that all the SGD updates for the mixture-specific parameters are of the form:

$$\lambda w_j(x)\Phi_j^{-1}e_j(x) \quad \lambda w_j(x)\begin{bmatrix} x - \mu \\ xx' - \Sigma - \mu\mu' \end{bmatrix}, \tag{23}$$

where the first line is for a general model, and the second for the Gaussian mixture. E.g., $w_j(x) = \phi_j\alpha_j(x)$ for the true objective or the ELBO, and $w_j(x) = \phi_j l_j(x)$ for our second Delta-method inspired algorithm. Due to the inverse of Σ in the natural parameters, the update rules involve a matrix inverse.

Upon statistical convergence, when SGD moves around a local optimum of the objective, we expect the average over observations of the updates above to be zero. This implies that $E[w_j(x)e_j(x)] = 0$ upon convergence, i.e., the weighted expected error per cluster is zero. For the Gaussian case, this statement, coupled with the specific form of the error in Equation 22, translates into

$$\sum_i w_j(x^{(i)})\mu_j^{(t)} = \sum_i w_j(x^{(i)})x^{(i)}, \quad \sum_i w_j(x^{(i)})\Sigma_j^{(t)} = \sum_i w_j(x^{(i)})\big(x^{(i)}x'^{(i)} - \mu_j^{(t)}\mu_j'^{(t)}\big). \tag{24}$$

A bit of algebra results in our well-known friends. Letting $n_j^t = \sum_{i=1}^t w_j(x^{(i)})$ be the effective number of observations coming from mixture j up to the first t observations, these are

$$\mu_j^{(t)} = \frac{1}{n_j^t}\sum_{i=1}^t w_j(x^{(i)})x^{(i)} \quad \Sigma_j^{(t)} = \frac{1}{n_j^t}\big(\sum_{i=1}^t w_j(x^{(i)})x^{(i)}x'^{(i)}\big) - \mu_j^{(t)}\mu_j'^{(t)}. \tag{25}$$

Notice that $w_j(x^{(i)})$ above is evaluated using the parameters at time $t-1$, i.e., would require re-evaluation of all weights at every time step. It is easy, however, to modify this computation to be online, simply by keeping old weights, and only evaluating the one corresponding to the new observation, i.e., performing the following updates:

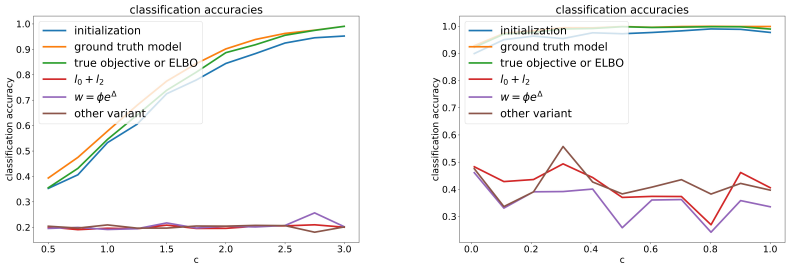
$$n_j^{t+1} = n_j^t + w_j(x^{(t+1)}), \quad \phi_j^{t+1} = n_j^{t+1}/\sum_{i=1}^k n_i^{t+1}, \quad \mu_j^{(t+1)} = \frac{1}{n_j^{t+1}}\left(n_j^t\mu_j^{(t)} + w_j(x^{(t+1)})x^{(t+1)}\right), \tag{26}$$

$$\Sigma_j^{(t+1)} = \frac{1}{n_j^{t+1}}\left(n_j^t\big(\Sigma_j^{(t)} + \mu_j^{(t)}\mu_j'^{(t)}\big) + w_j(x^{(t+1)})x^{(t+1)}x'^{(t+1)}\right) - \mu_j^{(t+1)}\mu_j'^{(t+1)}. \tag{27}$$

Plugging in t he specific form of $w_j(x)$ for each of our algorithm variants completes the learning algorithm. Of course, we also use the update equation for ϕ in the algorithms above to estimate the mixture proportions. In a way, then, the question is what is the best form of $w_j(x)$ to use. Finally, note that constant learning rates do not affect the steady state, and hence the update above is independent of λ. One can insert λ back simply redefining $w_j(x) = \lambda w_j(x)$, either with or without time-dependence in the learning rate.

6 Numerical Results

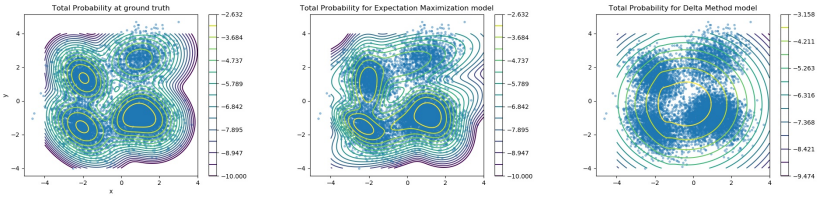
In our first numerical test, setup A, as ground truth (i.e., data generating model), we consider models with $K = 5$ clusters and $d = 2$, and parameters $\phi_j = 0.2$ for all j, $\mu_1 = [c, c]$, $\mu_2 = [-c, c]$, $\mu_3 = [-c, -c]$, $\mu_4 = [c, -c]$, and $\mu_5 = [00]$, and Σ_j with 1 on its diagonals, and correlations of -0.5, 0.5, -0.5, 0.5 and 0 for the 5 clusters. We control how hard the problem is by varying c. For small values compared to 1, the 5 cluster centers are close to overlapping, while for large values the various clusters are quite apart. In our second numerical test, we sample ground truth in each simulation as follows. After specifying K and d, and a constant c that controls cluster spread, we sample ϕ from a Dirichlet distribution with all parameters set to 1, μ_j from a zero-mean multivariate Gaussian with variance set to cI, where I is the identity, and Σ_j from an inverse Wishart distribution with scale equal to I, and $d + 2$ degrees of freedom. These distributions were chosen because they are the conjugates ones for the respective parameters. We show classification accuracy results in Figure 1 for the algorithms in Equation 27. We initialize them with a small sample of fully observed data using the MLE estimates. As expected, we see that using the ground truth model for classification is optimal, and that the accuracy of it increases with c. We also see that the accuracy is not too bad after the MLE initialization. The ELBO-based algorithm is as accurate as right after MLE initialization for small c, but does better, indeed as good as ground truth, as c increases. Unfortunately, we find that our new algorithms based on $\tilde{\ell}$ do not do well in the first two setups.



(a) Setup A. (b) Setup B with $K = 5$ and $d = 5$.

Figure 1: Classification accuracy using the various algorithms. The data sizes used were 50, 4000 and 1000 for initialization, training, and testing, respectively, in setup A, and 200, 10000 and a 1000 in setup B. The different algorithms relied on the same data. The ELBO-based algorithm used the ELBO as stopping criteria, and the others the marginal log likelihood. We also stopped them after two passes through the data regardless. For each value of c, we show the average of 6 simulations in setup A, and of 10 simulations in setup B.

In order to assess the qualitative reasons for the inaccuracy of the $l_0 + l_2$ approximation we visualized the fit $d = 2$. Here we initialize the means through sampling the multivariate normal distribution with a mean of $\vec{0}$ and an identity matrix times 3 as its variance. The covariance of each Gaussian is determined by sampling the Wichart distribution with a mean of $\vec{0}$ and the same covariance as above scaled by a factor in order to find a case where some of the Gaussians are separated and some are joined. 2 is a comparison graph between the true distribution and fits using EM and $\tilde{\ell}$. The SGD approach is done using the optimization through convergence criteria approach previously described.



(a) Ground Truth (b) True Objective and EM (c) $\tilde{\ell}(x)$ Model

Figure 2: Visual representation of the total fit through a probability contour map The contour is log scaled.

7 Conclusions

We developed an approximation to the true objective and used that approximation to develop a new algorithm for fitting FMM. We found that the true objective gradients are actually tractable. Because the E-step is tight and closed-form in these models, the ELBO in the M-step is identical to the true objective, so the ELBO and true objective gradients are the same. This allows for online algorithms based on these gradients that work well.

The new approximate objective we derived through the delta method results in algorithms that do not perform well. From the qualitative observation above, it perhaps reasonable to assume that a better approximation result may be achieved through more terms in the expansion, beyond $\ell_2(x)$,. However, in other models such as VAEs where the lower bound employed for variational inference is also not tight, a similar delta-method approach could prove competitive. We are less excited about this direction in light of the results here.

Contributions

All members contributed equally.

References

[1] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

[2] Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the Royal Statistical Society: Series B (Methodological) 39.1 (1977): 1-22.

[3] Amari, Shun-ichi. "Neural learning in structured parameter spaces-natural Riemannian gradient." Advances in neural information processing systems. 1997.

[4] Gómez-Uribe, Carlos Alberto, and Brian Karrer. "The decoupled extended Kalman filter for dynamic exponential-family factorization models." arXiv preprint arXiv:1806.09976 (2018).

[5] McLachlan, Geoffrey, and Thriyambakam Krishnan. The EM algorithm and extensions. Vol. 382. John Wiley & Sons, 2007.

[6] Puskorius, Gintaras V., and Lee A. Feldkamp. "Decoupled extended Kalman filter training of feedforward layered networks." Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on. Vol. 1. IEEE, 1991.

[7] Y. Ollivier. Online natural gradient as a kalman filter. arXiv preprint arXiv:1703.00209, 2017.

All python code that was created for this project is available here: https://github.com/vikrapivin/FMM_Project_CS229 . This code runs in the cs229 environment. We use python 3.7, numpy, scipy, and matplotlib.