CS731, Project, 2008-12-20, Nathanael Fillmore (nathanae)[1]

An EM-algorithm for non-parametric separable mixture models

1 Introduction

In this paper, we seek an efficient and theoretically justified algorithm to learn the parameters of a *separable mixture of kernel density estimators*; this model is detailed and motivated in §2. An efficient algorithm to learn the parameters already exists, namely Benaglia et al.'s heuristic ([1]; §4 below)—but it is not derived from any first principles, and it is not guaranteed to find a locally optimal solution. Our novel contributions are as follows:

- We derive a principled but slow expectation maximization (EM) algorithm to solve the problem (§3).
- We show that Benaglia et al.'s heuristic is an approximation to this EM algorithm (§4–§6).
- We propose an enhancment of Benaglia et al.'s heuristic that is principled and fast (§7).

We conclude with experimental results (§8) and discussion (§9).

Mixture models have been studied and applied extensively over the past decades; see the monographs [8, 9, 10]. Kernel density estimators have also been studied in detail; see, for example, [11]. The specific type of mixture model under consideration here is less well-studied, and there has been recent progress. In particular, Hall and others [6, 7] recently showed that the model is identifiable if the dimensionality of the data is large enough in comparison to the number of mixture components, and EM-like algorithms for learning the model's parameters have been proposed by [2, 3], in addition to [1].

2 Problem formulation

We seek to estimate a probability density function (pdf) $f(\mathbf{x})$ from n points of D-dimensional training data $\mathbf{x}_1, \ldots, \mathbf{x}_n$. We assume that $f(\mathbf{x})$ is a *mixture*, that is, a convex combination, of m components $f_j(\mathbf{x})$:

$$f(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j f_j(\mathbf{x}), \tag{1}$$

where all mixing weights $\lambda_j \geq 0$ and $\sum_{j=1}^{m} \lambda_j = 1$. Each component density $f_j(\mathbf{x})$ is *separable* as a product of D one-dimensional densities:

$$f_j(\mathbf{x}) = \prod_{d=1}^{D} f_{jd}(x_d) \tag{2}$$

[1]This report is based on joint work with Jingci Meng and Jerry Zhu. We have also talked to Steve Wright

In other words, the D features are conditionally independent. Each one-dimensional density f_{jd} is a *weighted kernel density estimator* (KDE) based on the training data $\mathbf{x}_1, \ldots, \mathbf{x}_n$:

$$f_{jd}(x_d) = \frac{1}{h} \sum_{i=1}^{n} \alpha_{ij} K\left(\frac{x_d - x_{id}}{h}\right) \tag{3}$$

where the KDE weight α_{ij} specifies how much the ith training datum should contribute to the jth KDE. As with $\boldsymbol{\lambda}$, we require that all $\alpha_{ij} \geq 0$ and that $\sum_{i=1}^{n} \alpha_{ij} = 1$ for all $j = 1, \ldots, m$. For the *kernel function* $K(\cdot)$, any pdf where $K(z) = K(-z)$ for all z in its domain can be used; in this paper we use the Gaussian kernel

$$K(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} z^2\right), \tag{4}$$

a common choice. Finally, h is the *bandwidth* of the KDE. A small bandwidth causes each kernel to be a pointy butte; large bandwidth causes plateaus. Ideally, the bandwidth will be neither too small nor too large. We assume that each kernel evaluation is relatively expensive, and that the number of training data n is too large to tractably store a precomputed kernel matrix $\bar{K}_{ii'd} = K\left(\frac{x_{id} - x_{i'd}}{h}\right)$.

We aim to find the "best" parameters $\Theta = \{\boldsymbol{\lambda}, \mathbf{w}\}$. (For simplicity we assume an appropriate bandwidth h is known.) A common and natural objective by which to judge the "goodness" of a particular choice of parameters is the *log likelihood*, defined for any training set $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and Θ-parameterized distribution P_Θ as $\ell(\Theta) = \sum_{i=1}^{n} \log P_\Theta(\mathbf{x}_i)$. Substituting our density f leads to the following concrete formula to calculate log likelihood:

$$\ell(\Theta) = \sum_{i=1}^{n} \log \frac{1}{h^D} \sum_{j=1}^{m} \lambda_j \prod_{d=1}^{D} \sum_{i'=1}^{n} \alpha_{i'j} K\left(\frac{x_{id} - x_{i'd}}{h}\right) \tag{5}$$

To find the parameters Θ that maximize log likelihood, we solve the following optimization problem:

$$\begin{array}{lll} \max\limits_{\Theta} & \ell(\Theta) & \\ \text{s.t.} & \sum_{j=1}^{m} \lambda_j = 1 & \\ & \sum_{i=1}^{n} \alpha_{ij} = 1 & \text{for} \quad j = 1, \ldots, m \\ & \lambda_j \geq 0 & \text{for} \quad j = 1, \ldots, m \\ & \alpha_{ij} \geq 0 & \text{for} \quad i = 1, \ldots, n,\ j = 1, \ldots, m \end{array} \tag{6}$$

3 EM algorithm: principled but slow

One way to solve problem (6) is to use an expectation-maximization (EM) algorithm [5]. EM has proved quite effective for learning the parameters of many mixture models (e.g. [8] chs. 2 and 12). The EM algorithm cycles between two steps, one (the *E step*) which calculates the expected value of a hidden variable, and one (the *M step*) which finds the best parameters, given that expected value. Ideally, by providing a concrete value for the hidden variable, the M-step objective will be simpler than the overall objective—in the best case it will have a closed-form maximizer. A key

property of the EM algorithm is that it is guaranteed (a) to increase the original objective at each iteration, and (b) to converge to a local maximum of the objective.

To derive an EM algorithm for problem (6), we introduce a distribution $P_q(j|\mathbf{x}_i)$, as follows. First, for convenient manipulation we will write our target pdf $f(\mathbf{x}_i)$ as a generic probability distribution parameterized by Θ, that is, as $P_\Theta(\mathbf{x}_i)$. With this notation, the log likelihood is expressed as

$$\ell(\Theta) = \sum_{i=1}^{n} \log P_\Theta(\mathbf{x}_i) \tag{7}$$

as before. Next, we introduce a new discrete probability distribution $P_q(j|\mathbf{x}_i)$, parameterized by an $n \times m$ matrix $[q_{ij}]$, where each row represents the conditional probability that datum $\mathbf{x}_i$ was drawn from component j. Since any discrete probability distribution sums to 1, $\sum_{j=1}^{m} P_q(j|\mathbf{x}_i) = 1$ for all i, and our log likelihood can be trivially rewritten as

$$\ell(\Theta) = \sum_{i=1}^{n} \underbrace{\sum_{j=1}^{m} P_q(j|\mathbf{x}_i)}_{=1} \log P_\Theta(\mathbf{x}_i) \tag{8}$$

Finally, we introduce a joint distribution, again parameterized by Θ, namely

$$P_\Theta(\mathbf{x}_i, j) = P_\Theta(j) P_\Theta(\mathbf{x}_i|j) = \lambda_j f_j(\mathbf{x}_i) \tag{9}$$

and perform technical manipulations to find a useful decomposition of the log likelihood into Kullback-Leibler (KL) divergence and a new objective function $F(\Theta; q)$:

$$\begin{aligned}
\ell(\Theta) &= \sum_{i=1}^{n}\sum_{j=1}^{m} P_q(j|\mathbf{x}_i) \log \left[\frac{P_\Theta(\mathbf{x}_i) P_q(j|\mathbf{x}_i)}{P_\Theta(\mathbf{x}_i, j)} \frac{P_\Theta(\mathbf{x}_i, j)}{P_q(j|\mathbf{x}_i)} \right] && (10)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} P_q(j|\mathbf{x}_i) \log \frac{P_\Theta(\mathbf{x}_i) P_q(j|\mathbf{x}_i)}{P_\Theta(\mathbf{x}_i, j)} + \sum_{i=1}^{n}\sum_{j=1}^{m} P_q(j|\mathbf{x}_i) \log \frac{P_\Theta(\mathbf{x}_i, j)}{P_q(j|\mathbf{x}_i)} && (11)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} P_q(j|\mathbf{x}_i) \log \frac{P_\Theta(\mathbf{x}_i) P_q(j|\mathbf{x}_i)}{P_\Theta(\mathbf{x}_i) P_\Theta(j|\mathbf{x}_i)} + \sum_{i=1}^{n}\sum_{j=1}^{m} P_q(j|\mathbf{x}_i) \log \frac{P_\Theta(\mathbf{x}_i, j)}{P_q(j|\mathbf{x}_i)} && (12)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} P_q(j|\mathbf{x}_i) \log \frac{P_q(j|\mathbf{x}_i)}{P_\Theta(j|\mathbf{x}_i)} + \sum_{i=1}^{n}\sum_{j=1}^{m} P_q(j|\mathbf{x}_i) \log \frac{P_\Theta(\mathbf{x}_i, j)}{P_q(j|\mathbf{x}_i)} && (13)\\
&= \left[\sum_{i=1}^{n} \mathrm{KL}(P_q(\cdot|\mathbf{x}_i) || P_\Theta(\cdot|\mathbf{x}_i)) \right] + F(\Theta; q) && (14)
\end{aligned}$$

The KL divergence, defined for discrete distributions P_a and P_b by

$$KL(P_a||P_b) = \sum_{i=1}^{n} P_a(x_i) \log(P_a(x_i)/P_b(x_i)) \tag{15}$$

is (i) always non-negative and (ii) is zero only when $P_a = P_b$. Due to these properties, F is a lower bound on the log likelihood, and F *equals* the log likelihood when $P_q = P_\Theta$, since in that case the first term of Eq. (14) is zero.

This leads to an EM algorithm. In the E step, we will compute the parameter q of $P_q(j|\mathbf{x}_i)$ for all i and j, based on the current information in Θ. In the M step, we will use this q to find the Θ that maximizes $F(\Theta; q)$. By looping between these steps, we are guaranteed to converge monotonically to a local maximum of the original log likelihood function.

Concretely, in the E step, we will set q_{ij} so that

$$q_{ij} \equiv P_q(j|\mathbf{x}_i) = P_\Theta(j|\mathbf{x}_i) \tag{16}$$

i.e., to minimize KL divergence. By Bayes' rule,

$$P_\Theta(j|\mathbf{x}_i) = P_\Theta(j) P_\Theta(j|\mathbf{x}_i) / P_\Theta(\mathbf{x}_i) \tag{17}$$

Substituting for P_Θ in the right-hand side of (17), we obtain our update:

$$q_{ij} = \frac{\lambda_j \prod_{d=1}^{D} f_{jd}(x_{id})}{\sum_{j'=1}^{n} \lambda_{j'} \prod_{d=1}^{D} f_{j'd}(x_{id})} \tag{18}$$

In the M step, we will maximize $F(\Theta; q)$ over Θ subject to the constraints of problem (6). Substituing for F, P_q and P_Θ, we have

$$F(\Theta; q) = \sum_{i=1}^{n} \sum_{j=1}^{m} P_q(j|\mathbf{x}_i) \log \frac{P_\Theta(\mathbf{x}_i, j)}{P_q(j|\mathbf{x}_i)} \tag{19}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{m} q_{ij} \log \frac{\lambda_j f_j(\mathbf{x}_i)}{q_{ij}} \tag{20}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{m} [q_{ij} \log \lambda_j f_j(\mathbf{x}_i) - q_{ij} \log q_{ij}] \tag{21}$$

Since the second term of (21) does not include $\boldsymbol{\lambda}$ or $\boldsymbol{\alpha}$, we can ignore it while optimizing:

$$F(\Theta; q) = \sum_{i=1}^{n} \sum_{j=1}^{m} q_{ij} \log \lambda_j f_j(\mathbf{x}_i) + \text{const} \tag{22}$$

Finally, we substitute for f_j and rearrange:

$$F(\Theta; q) = \sum_{i=1}^{n} \sum_{j=1}^{m} q_{ij} \log \left[\lambda_j \prod_{d=1}^{D} \frac{1}{h} \sum_{i'=1}^{n} \alpha_{i'j} K\left(\frac{x_{id} - x_{i'd}}{h} \right) \right] + \text{const} \tag{23}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{m} q_{ij} \left[\log \lambda_j + \sum_{d=1}^{D} \log \frac{1}{h} \sum_{i'=1}^{n} \alpha_{i'j} K\left(\frac{x_{id} - x_{i'd}}{h} \right) \right] + \text{const} \tag{24}$$

Note that the objective Eq. (24) is separable into a part involving $\boldsymbol{\lambda}$ and a part involving $\boldsymbol{\alpha}$. These can be optimized separately. To find the optimal $\boldsymbol{\lambda}$ we formulate the Lagrangian:

$$\Lambda(\boldsymbol{\lambda}, \beta) = \sum_{i=1}^{n} \sum_{j=1}^{m} q_{ij} \log \lambda_j + \beta \left[\sum_{j=1}^{m} \lambda_j - 1 \right] \tag{25}$$

Next we compute the gradient:

$$\frac{\partial \Lambda}{\partial \lambda_j} = \frac{\sum_{i=1}^n q_{ij}}{\lambda_j} + \beta \tag{26}$$

$$\frac{\partial \Lambda}{\partial \beta} = \sum_{j=1}^m \lambda_j - 1 \tag{27}$$

Setting the gradient to zero and solving for λ_j leads to the following closed-form solution:

$$\lambda_j^{t+1} = \frac{1}{n} \sum_{i=1}^n q_{ij} \tag{28}$$

Unfortunately the situation with $\boldsymbol{\alpha}$ is not so nice. The Lagrangian is

$$\Lambda(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = \sum_{i=1}^n \sum_{j=1}^m q_{ij} \sum_{d=1}^D \log \frac{1}{h} \sum_{i'=1}^n \alpha_{i'j} K\left(\frac{x_{id} - x_{i'd}}{h}\right) + \sum_{j=1}^m \gamma_j \left[\sum_{i=1}^n \alpha_{ij} - 1\right] \tag{29}$$

The gradient is

$$\frac{\partial \Lambda}{\partial \alpha_{kj}} = \sum_{i=1}^n q_{ij} \sum_{d=1}^D \left[\frac{1}{\frac{1}{h}\sum_{k'=1}^n \alpha_{k'j} K\left(\frac{x_{id} - x_{k'd}}{h}\right)} \cdot \frac{1}{h} K\left(\frac{x_{id} - x_{kd}}{h}\right)\right] + \gamma_j \tag{30}$$

$$= \sum_{i=1}^n q_{ij} \sum_{d=1}^D \left[\frac{K\left(\frac{x_{id} - x_{kd}}{h}\right)}{\sum_{k'=1}^n \alpha_{k'j} K\left(\frac{x_{id} - x_{k'd}}{h}\right)}\right] + \gamma_j \tag{31}$$

$$\frac{\partial \Lambda}{\partial \gamma_j} = \sum_{i=1}^n \alpha_{ij} - 1 \tag{32}$$

There is no closed form update of $\boldsymbol{\alpha}$. However, the objective is concave in $\boldsymbol{\alpha}$, since it is a nonnegative weighted sum of the log of a linear function of $\boldsymbol{\alpha}$ ([4], chapter 3). Therefore we will use a gradient projection line search algorithm [12] to find the globally optimal $\boldsymbol{\alpha}$ in the simplex where the constraints of problem (6) are satisfied, that is, where all α_{ij} are nonnegative and $\sum_{i=1}^n \alpha_{ij} = 1$ for all $j = 1, \ldots, m$. In other words, we will take a step in the direction of the gradient, project into the feasible set, and repeat until convergence.[2]

In sum, we have the following algorithm:

Algorithm 1 (True EM)**.** Repeat for $t = 0, 1, 2, \ldots$ until convergence:

1. E step: for all i, j, set

$$q_{ij}^t = \frac{\lambda_j^t \prod_{d=1}^D f_{jd}^t(x_{id})}{\sum_{j'=1}^m \lambda_{j'}^t \prod_{d=1}^D f_{j'd}^t(x_{id})} \tag{40}$$

[2]In practice, the gradient with respect to $\boldsymbol{\alpha}$ as written above is numerically unstable. We avoid this problem by

2. M step: For $\boldsymbol{\lambda}$, there is a closed form maximizer. For all j, set

$$\lambda_j^{t+1} = \frac{1}{n}\sum_{i=1}^{n} q_{ij}^t \tag{41}$$

For $\boldsymbol{\alpha}$, we use backtracking line search with gradient projection to find the maximizer. Initially, let step size $s = 1$, and choose $\rho = 1/2 \in (0, 1)$. Repeat:

(a) For all i, j, set

$$\bar{\alpha}_{ij} = \alpha_{ij}^t + s\,(\partial F/\partial \alpha_{ij}) \tag{42}$$

$$\hat{\alpha}_{ij} = \frac{\bar{\alpha}_{ij}}{\sum_{i=1}^{n} \bar{\alpha}_{ij}} \tag{43}$$

(b) If $F(\hat{\boldsymbol{\alpha}}) \geq F(\boldsymbol{\alpha}^t)$, terminate and set $\boldsymbol{\alpha}^{t+1} = \hat{\boldsymbol{\alpha}}$. Otherwise, set $s = s\,\rho$.

□

Unfortunately, as mentioned above and shown empirically by our experiments (§8), the inner line search makes this EM algorithm as a whole extremely inefficient.

4 Benaglia et al.'s heuristic: fast but unprincipled

Benaglia et al. [1] propose the following "pseudo-EM" heuristic algorithm to find the parameters Θ. The heuristic is based on a slightly different model than the one we're interested in. Instead

---

substituting for q_{ij} and cancelling:

$$\frac{\partial \Lambda}{\partial \alpha_{kj}} = \sum_{i=1}^{n} q_{ij} \sum_{d=1}^{D}\left[\frac{K_{ikd}}{\sum_{k'=1}^{n}\alpha_{k'j}K_{ik'd}}\right] + \gamma_j \tag{33}$$

$$= \sum_{i=1}^{n} \frac{\lambda_j \prod_{d=1}^{D} f_{jd}(x_{id})}{\sum_{j'=1}^{n}\lambda_{j'}\prod_{d=1}^{D} f_{j'd}(x_{id})} \sum_{d=1}^{D}\left[\frac{K_{ikd}}{\sum_{k'=1}^{n}\alpha_{k'j}K_{ik'd}}\right] + \gamma_j \tag{34}$$

$$= \sum_{i=1}^{n} \frac{\lambda_j \prod_{d=1}^{D} f_{jd}(x_{id})}{\sum_{j'=1}^{n}\lambda_{j'}\prod_{d=1}^{D} f_{j'd}(x_{id})} \sum_{d=1}^{D}\left[\frac{K_{ikd}}{h f_{jd}(x_{id})}\right] + \gamma_j \tag{35}$$

$$= \sum_{i=1}^{n} \frac{\lambda_j \prod_{d=1}^{D} f_{jd}(x_{id})}{f(\mathbf{x}_i)} \sum_{d=1}^{D}\left[\frac{K_{ikd}}{h f_{jd}(x_{id})}\right] + \gamma_j \tag{36}$$

$$= \sum_{i=1}^{n}\sum_{d=1}^{D} \frac{\lambda_j \prod_{d'=1}^{D} f_{jd'}(x_{id'})}{f(\mathbf{x}_i)} \frac{K_{ikd}}{h f_{jd}(x_{id})} + \gamma_j \tag{37}$$

$$= \sum_{i=1}^{n}\sum_{d=1}^{D} \frac{\lambda_j \prod_{d'\neq d} f_{jd'}(x_{id'})}{h f(\mathbf{x}_i)} K_{ikd} + \gamma_j \tag{38}$$

$$= \sum_{i=1}^{n} \frac{\lambda_j}{h f(\mathbf{x}_i)} \sum_{d=1}^{D} K_{ikd} \prod_{d'\neq d} f_{jd'}(x_{id'}) + \gamma_j \tag{39}$$

of column-normalized data weights $\boldsymbol{\alpha}$, the Benaglia et al. use row-normalized weights $\mathbf{w}$. That is, the component KDE $f_{jd}(x_{id})$ is replaced by $\bar{f}_{jd}(x_{id})$, defined as

$$\bar{f}_{jd}(x_{id}) = \frac{1}{h} \sum_{i'=1}^{n} \left[\frac{w_{i'j}}{\sum_{i'=1}^{n} w_{i'j}} \right] K\left(\frac{x_{id} - x_{i'd}}{h} \right) \tag{44}$$

As Eq. (44) suggests, we can translate between $\boldsymbol{\alpha}$ and $\mathbf{w}$ using

$$w_{ij} = \frac{\alpha_{ij}}{\sum_{j'} \alpha_{ij'}} \quad \text{and} \quad \alpha_{ij} = \frac{w_{ij}}{\sum_{i'} w_{i'j}}, \tag{45}$$

that is, by renormalizing. The heuristic is as follows:

Algorithm 2 (heuristic)**.** Repeat for $t = 0, 1, 2, \ldots$ until convergence:

1. "E step": for all i, j, set

$$w_{ij}^{t} = \frac{\lambda_j^t \prod_{d=1}^{D} f_{jd}^t(x_{id})}{\sum_{j'=1}^{m} \lambda_{j'}^t \prod_{d=1}^{D} f_{j'd}^t(x_{id})} \tag{46}$$

2. "M step": for all j, set

$$\lambda_j^{t+1} = \frac{1}{n} \sum_{i=1}^{n} w_{ij}^t \tag{47}$$

For $t = 0$, Benaglia et al. skip the "E step" and instead use k-means clustering to obtain w_{ij}^0. □

As our experimental results indicate (§8), this heuristic is efficient and typically converges to a result near—though not quite at—a locally optimal parameter set Θ. In a way, it makes intuitive sense that the heuristic works. Observe that we can interpret λ_j as $P_\Theta(j)$ (the prior probability of component j), and $\prod_{d=1}^{D} f_{jd}(x_{id})$ as $P_\Theta(\mathbf{x}_i|j)$ (the conditional probability of datum $\mathbf{x}_i$ given component j). Then the w_{ij} update can be interpreted as

$$\frac{P_\Theta(j) P_\Theta(\mathbf{x}_i|j)}{\sum_{j'=1}^{m} P_\Theta(j') P_\Theta(\mathbf{x}_i|j')} = \frac{P_\Theta(j) P_\Theta(\mathbf{x}_i|j)}{P_\Theta(\mathbf{x}_i)} = P_\Theta(j|\mathbf{x}_i) \tag{48}$$

exactly as it ought. The update to the mixing weight λ_j also has a simple—and very reasonable—interpretation as the fraction of training data points we currently believe were drawn from component j, where we allow data to be split fractionally among several components.

Unfortunately, there's not much else to say about this algorithm. In particular, there is no theoretical guarantee that it will converge to a local optimum, or even get close. In practice, the algorithm does seems to converge quickly to a good result—but why?

5 "Lemma": Benaglia et al.'s heuristic in terms of q and $\boldsymbol{\alpha}$

We will be able to answer this question shortly, but first, to simplify the analysis, we reformulate Benaglia et al.'s heuristic so that instead of $\mathbf{w}$ it uses (i) q and (ii) $\boldsymbol{\alpha}$. This can be done as follows. (i) First, observe that the heuristic's "E-step" update (Eq. 46) is identical to the true E-step update (Eq. 40), except that the former updates $\mathbf{w}$ and the latter updates q. Similarly note that the heuristic's "M-step" update to $\boldsymbol{\lambda}$ (Eq. 47) is exactly the same as the true update (Eq. 41), except one sums over $\mathbf{w}$ and one over q. (ii) On the other hand, we already observed (Eq. 45) that $w_{ij} / \sum_{i'=1}^{n} w_{i'j} = \alpha_{ij}$.

This leads to the following algorithm, which substitutes q for $\mathbf{w}$ and f_{jd} for $\bar{f}_{jd}$, and sets $\boldsymbol{\alpha}$ at each iteration to the column-normalized q:

Algorithm 3. Repeat for $t = 0, 1, 2, \ldots$ until convergence:

1. E step: for all i, j, set

$$q_{ij}^t = \frac{\lambda_j^t \prod_{d=1}^{D} f_{jd}^t(x_{id})}{\sum_{j'=1}^{m} \lambda_{j'}^t \prod_{d=1}^{D} f_{j'd}^t(x_{id})} \tag{49}$$

2. M step: for all i, j, set

$$\lambda_j^{t+1} = \frac{1}{n} \sum_{i=1}^{n} q_{ij}^t \tag{50}$$

$$\alpha_{ij}^{t+1} = \frac{q_{ij}^t}{\sum_{i'} q_{i'j}^t} \tag{51}$$

For $t = 0$, we skip the E step and instead use k-means clustering to obtain q_{ij}^0. □

On one hand, this algorithm is equivalent to Benaglia et al.'s heuristic, in the sense that at each iteration the parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$ (or renormalized $\mathbf{w}$) will match between the two algorithms. On the other hand, this algorithm has the same superficial form (though not the same properties) as the true EM algorithm (Alg. 1), apart from the update to $\boldsymbol{\alpha}$.

This reformulation makes clear just what the shortcoming to Benaglia et al.'s algorithm is: although as we saw above in §4, Benaglia et al.'s update to w_{ij} is intuitively sensible, it may not increase log likelihood. The update to $\boldsymbol{\lambda}$, on the other hand, is correct in Benaglia et al.'s algorithm. So the key thing to focus on is the update of $\boldsymbol{\alpha}$.

6 Benaglia et al. as EM with a delta kernel

We are now able to show why Benaglia et al.'s algorithm usually works: it is an approximation to a degenerate EM algorithm.

To see this, let's look at what happens as the bandwidth h approaches zero. In this case, each kernel becomes a delta function above $\mathbf{x}_{id}$:

$$\lim_{h \to 0} K\left(\frac{\mathbf{x}_{id} - \mathbf{x}_{i'd}}{h}\right) = \begin{cases} \infty & i = i' \\ 0 & \text{otherwise} \end{cases} \tag{52}$$

Next let's examine what happens to the gradient of the M-step Lagrangian in $\boldsymbol{\alpha}$ as this happens. Recall the Lagrangian's gradient:

$$\frac{\partial \Lambda}{\partial \alpha_{kj}} = \sum_{i=1}^{n} q_{ij}^t \sum_{d=1}^{D} \left[\frac{K\left(\frac{x_{id}-x_{kd}}{h}\right)}{\sum_{i'=1}^{n} \alpha_{i'j} K\left(\frac{x_{id}-x_{i'd}}{h}\right)} \right] + \gamma_j \tag{53}$$

In the limit, all the sums over i and i' disappear, since only $K(\frac{x_{kd}-x_{kd}}{h}) = K(0)$ has support. That is, we have

$$\lim_{h \to 0} \frac{\partial \Lambda}{\partial \alpha_{kj}} = q_{kj}^t \sum_{d=1}^{D} \left[\frac{K(0)}{\alpha_{kj} K(0)} \right] + \gamma_j \tag{54}$$

$$= q_{kj}^t \sum_{d=1}^{D} \left[\frac{1}{\alpha_{kj}} \right] + \gamma_j \tag{55}$$

$$= q_{kj}^t D/\alpha_{kj} + \gamma_j \tag{56}$$

Setting this gradient to zero and solving for α_{kj}, we obtain Bengalia et al.'s update (Eq. 51):

$$\alpha_{kj}^{t+1} = \frac{q_{kj}^t}{\sum_{i'=1}^{n} q_{i'j}^t} \tag{57}$$

This result explains why the heuristic's update to $\boldsymbol{\alpha}$ generally works: in the special case of a delta kernel, it is the correct update. Of course, in practice, a delta kernel cannot be used—severe overfitting would result—and in that case, as we have observed, the update is not guaranteed to move in a direction of ascent.

7 Generalized EM: fast *and* principled

There is a simple but effective way to fix Benaglia et al.'s heuristic so that it is guaranteed to increase log likelihood at each iteration. As in true EM, we do a backtracking line search to update $\boldsymbol{\alpha}$, but we search on a line that interpolates between the heuristic update and the gradient.

Algorithm 4 (GEM)**.** Repeat for $t = 0, 1, 2, \ldots$ until convergence:

1. E step: for all i, j, set

$$q_{ij}^t = \frac{\lambda_j^t \prod_{d=1}^{D} f_{jd}^t(x_{id})}{\sum_{j'=1}^{m} \lambda_{j'}^t \prod_{d=1}^{D} f_{j'd}^t(x_{id})} \quad \text{(same as Alg. 1)} \tag{58}$$

2. M step: for all j, set

$$\lambda_j^{t+1} = \frac{1}{n} \sum_{i=1}^{n} q_{ij}^t \quad \text{(same as Alg. 1)} \tag{59}$$

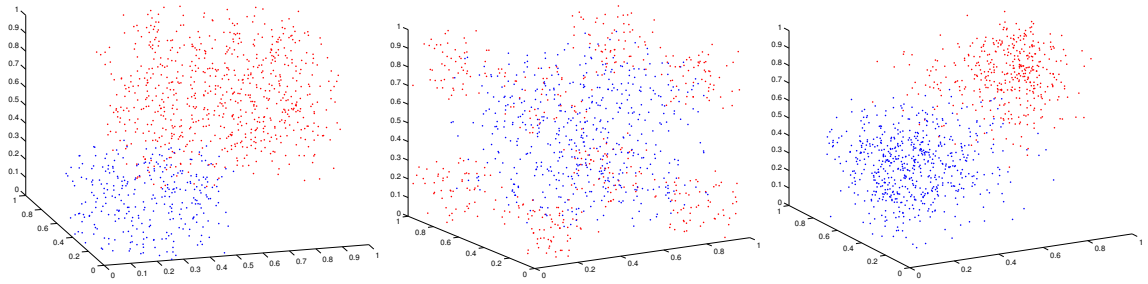

Figure 1: From left to right: the unifs, holy, and betas datasets. Component 1 is in blue; component 2 is in red.

For $\boldsymbol{\alpha}$, we use backtracking line search on a line between the heuristic and steepest-ascent steps. Initially, we compute the heuristic update

$$\tilde{\alpha}_{ij} = \frac{q_{ij}^t}{\sum_{i'} q_{i'j}^t}, \tag{60}$$

and let $\mathbf{p} = \tilde{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^t$ denote the step from the current $\boldsymbol{\alpha}$ to the heuristic $\boldsymbol{\alpha}$. Let step size $s = 1$, and choose $\rho = 1/2 \in (0, 1)$. Repeat:

(a) For all i, j, set

$$\bar{\alpha}_{ij} = \alpha_{ij}^t + s\,(s\,p_{ij} + (1 - s)\,(\partial F / \partial \alpha_{ij})) \tag{61}$$

$$\hat{\alpha}_{ij} = \frac{\bar{\alpha}_{ij}}{\sum_{i=1}^n \bar{\alpha}_{ij}} \tag{62}$$

(b) If $F(\hat{\boldsymbol{\alpha}}) \geq F(\boldsymbol{\alpha}^t)$, terminate and set $\boldsymbol{\alpha}^{t+1} = \hat{\boldsymbol{\alpha}}$. Otherwise, set $s = s\,\rho$.

□

During the first iteration of the inner line search above, $s = 1$, so $\bar{\alpha}_{ij} = \alpha_{ij}^t + p_{ij} = \tilde{\alpha}_{ij}$, the heuristic update. Since the heuristic update usually works, Alg. 4 will usually be fast. When the heuristic doesn't work, the inner loop will search along a line that is tangent to the gradient near $\boldsymbol{\alpha}^t$. This guarantees that, except at a local maximum, Alg. 4 will always find and take an ascent step—so Alg. 4 is principled. Since we do not maximize the M-step objective F in $\boldsymbol{\alpha}$, Alg. 4 is only generalized EM, not full EM. Nevertheless, like full EM, it is guaranteed to converge monotonically to a local optimum.

8 Experiments

We ran experiments on the following synthetic datasets, all lying in $[0, 1]^3$:

"unifs", a simple mixture of two uniforms: For $i = 1, \ldots, n$, we sample $j = 1$ with probability $3/10$ and $j = 2$ otherwise. If $j = 1$, we sample $x_{id} \sim \text{uniform}[0, 1/2]$ for $d = 1, \ldots, 3$. If $j = 2$, we sample $x_{id} \sim \text{uniform}[1/4, 1]$ for $d = 1, \ldots, 3$.

		EM				heuristic			GEM					truth
dataset	n	iters	time	iters/ls	ℓ	iters	time	ℓ	iters	time	iters/ls	ls	ℓ	ℓ
unifs	10	7	0.60s	58	20.19	15	0.06s	23.17	19	0.32s	57	1	23.17	21.66
	50	1492	1663s	280	18.64	143	1.4s	38.86	28	15s	402	9	37.85	38.64
	100	126	2087s	1059	19.40	14	0.40s	65.12	19	23s	234	6	65.11	65.30
holy	10	152	1.8s	2	14.99	25	0.06s	20.03	14	0.15s	56	1	20.03	14.94
	50	>3775	>2h		>21.34*	276	2.8s	34.67	267	4.0s	63	1	34.67	32.71
	100	>727	>2h		>34.54*	29	0.58s	51.77	472	93s	737	6	51.61	51.28
betas	10	6	0.52s	59	19.63	45	0.09s	21.59	44	1.77s	1074	1	21.59	21.59
	50	10	23s	569	39.15	60	0.36s	50.27	13	17s	1074	1	50.26	49.78
	100	9	133s	963	80.16	8	0.17s	97.17	16	38s	401	6	97.12	95.72

* This test did not finish in a reasonable amount of time; when I killed the job, the log likelihood had plateaued for hundreds of iterations at the indicated value.

Table 1: Summary of experimental results. See text for details.

"holy", a mixture where the first component spans a hole left by the second component; the effect is that component 2 consists of nine cubes while component 1 consists of background noise: For $i = 1, \ldots, n$, we sample $j = 1$ with probability $1/2$ and $j = 2$ otherwise. If $j = 1$, we sample $x_{id} \sim$ uniform$[0.1, 0.9]$, for $d = 1, \ldots, 3$. If $j = 2$, we sample $x_{id} \sim$ uniform$[0.0, 0.3] \cup [0.7, 1.0]$, for $d = 1, \ldots, 3$.

"betas", a mixture of two beta-distributed components with different parameterizations in different dimensions; the effect is that the components are less well-separated in dimension 3 than in dimension 1: For $i = 1, \ldots, n$, we sample $j = 1$ with probability $3/5$ and $j = 2$ otherwise. If $j = 1$, we sample $x_{id} \sim$ beta$(d, 4)$, for $d = 1, \ldots, 3$. If $j = 2$, we sample $x_{id} \sim$ beta$(6, d)$, for $d = 1, \ldots, 3$.

These datasets are plotted in Fig. 1. From each dataset, we drew samples of $n = 10$, 50, and 100 points for our experiments.

We ran true EM (Alg. 1), Benaglia et al.'s heuristic (Alg. 2), and GEM (Alg. 4) on each sample, starting in all cases from a k-means clustering. Bandwidth of $h = 0.05$ was chosen based on an informal analysis of plots. For all algorithms, we recorded the total number of EM iterations required (the "iters" column), the total time required ("time"), and the ending log likelihood ("ℓ"). For true EM, we also recorded the average number of iterations required in the line search ("iters/ls"). For GEM, we recorded the number of times the line search was required ("ls") and the average number of iterations per line search ("iters/ls"). Finally, we also computed the log likelihood of the "true" parameters, i.e. the parameters based on the true component labels of each datum. Results are summarized in Table 1.

The experiments demonstrate several key points—and raise further questions. First, as already asserted several times above, the time required by Benaglia et al.'s heuristic is much less than the time required by EM, especially as n increases. This is due to the expense of computing the gradient and performing an inner line search every iteration.

Second, despite its lack of a theoretical convergence guarantee, the heuristic actually finds a *better*—sometimes much better—solution than EM in all our experiments. Although we do not yet know precisely why this happens, it is clear why it *can* happen: the log likelihood is highly nonconvex, so the local optimum found by gradient ascent is not likely to be the best optimization technique available. The "ls" column of Table 1 is never zero; this indicates that the heuristic is indeed making "unjustified" steps that temporarily decrease the log likelihood, but ultimately

may lead to a better solution.

Third, GEM finds a solution with nearly equal log likelihood to the heuristic, while maintaining a theoretical convergence guarantee. In other words, GEM is not just theoretically successful—it's also empirically successful at converging quickly to a good solution. This makes sense, since GEM usually takes the same step as the heuristic, and even when it doesn't, GEM tries to at least stay close to the heuristic's suggestion while still increasing log likelihood monotonically.

Finally, note that sometimes the solution found by the heuristic (or GEM) has greater log likelihood than that of the "true" parameter set. This can happen if one of the training data points is drawn from (say) component 1, in a region where component 1's true pdf is lower than component 2's pdf. The "true" parameters would assign this point to component 1, but assigning it to component 2 would increase log likelihood.

9 Conclusion

We have made progress analyzing—and improving upon—Benaglia et al.'s heuristic, but much fascinating work remains. In particular, we intend (a) to seek a deeper understanding as to why Benaglia et al.'s heuristic works, (b) to explore different techniques (perhaps using second-order information) for finding the global optimum, and (c) to perform experiments on larger and more varied datasets.

References

[1] Tatiana Benaglia, Didier Chauveau, and David R. Hunter. An em-like algorithm for semi- and nonparametric estimation in multivariate mixtures. December 2007.

[2] Gregory Beylkin, Jochen Garcke, and Martin J. Mohlenkamp. Multivariate regression and machine learning with sums of separable functions. 2007.

[3] Laurent Bordes, Didier Chauveau, and Pierre Vandekerkhove. An em algorithm for a semiparametric mixture model. 2006.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[6] Peter Hall, Amnon Neeman, Reza Pakyari, and Ryan Elmore. Nonparametric inference in multivariate mixtures. *Biometrika*, 92(3):667–678, September 2005.

[7] Peter Hall and Xiao-Hua Zhou. Nonparametric estimation of component distributions in a multivariate mixture. *The Annals of Statistics*, 31(1):201–224, 2003.

[8] Geoffrey McLachlan and David Peel. *Finite mixture models*. Wiley, 2000.

[9] B.W. Silverman. *Density estimation for statistics and data analysis*. 1986.

[10] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical analysis of finite mixture distributions*. 1985.

[11] Larry Wasserman. *All of Nonparametric Statistics*. Springer, 2007.

[12] Stephen J. Wright. Gradient projection handout, 2008. http://www.cs.wisc.edu/∼swright/726/handouts/gradient-projection-2008.pdf.