2018 IEEE International Conference on Data Mining

LEEM: Lean Elastic EM for Gaussian Mixture Model via Bounds-Based Filtering

Shuai Yang, Xipeng Shen
Department of Computer Science
North Carolina State University, Raleigh, USA
{*syang16,xshen5*}*@ncsu.edu*

***Abstract*—Gaussian Mixture Model (GMM) is widely used in characterizing complicated real-world data and has played a crucial role in many pattern recognition problems. GMM is usually trained by Expectation Maximization algorithm (EM) which is computationally intensive. Previous studies have proposed a family of variants of EM. By considering only the data points that are the most important to a model in a GMM when updating that model, they help reduce some GMM training time. They are named Elastic EM in this paper. This work proposes several novel optimizations to further accelerate Elastic EM. These optimizations detect and avoid unnecessary probability calculations through novel bounds-based filtering at E-step as well as a Delta optimization to the M-step. Together, they create Lean Elastic EM (LEEM), which brings multi-fold speedups on six datasets of various sizes and dimensions.**

***Keywords*-Gaussian Mixture Model, Acceleration, Expectation Maximization, Elastic EM**

I. Introduction

GMM is an important probabilistic model with multiple normally distributed subpopulations (mixture components) within an overall population. GMM is powerful in representing arbitrarily complex distributions with multiple mixture components. It has been applied to model various data with multi-modality in nature, from speech signals to textures, images, and so on. GMM has been long time served as the key component of speech recognition system [1]–[3]. In a seminal work [1], a GMM corresponds to one speaker and individual mixture component of GMM represents some of that speaker's spectral features that are effective for identifying the speaker. GMM has been applied for recognizing different languages [2], diverse dialects [4] and speaker emotions [3]. Besides speech, GMM has been applied to computer vision [5], remote sensing image processing [6], medical images [7], and so on.

Although Deep Neural Networks (DNN) have been introduced to the related fields [8], [9] in recent years, compared to DNN, GMM has some important appealing properties: less training data required, producing not only classification but also the distribution models of each category offering insights and interpretability of the observations, much fewer hyperparameters to tune, and so on. So despite the recent rise of DNN, GMM remains an important approach to data mining and machine learning.

One GMM is composed by multiple multivariate Gaussian distributions. Training a GMM is to find the maximum likelihood estimation (MLE) of parameters of all multivariate Gaussian distributions. Yet there is no closed-form solutions to MLE of GMM. EM algorithm [10] hence is used to approximate the MLE of GMM. EM algorithm, also known as Soft EM, is an iterative method that maximizes likelihood estimation of parameters. After each iteration, the likelihood of observation increases until convergence.

Although EM algorithm has been serving as the standard tool to train GMM, MLE via EM algorithm can be very time-consuming. Calculations in both E-step and M-step are inherently expensive, especially when dealing with large datasets and high-dimensional data. The needs for many iterations to converge worsen the issue. A number of studies have attempted to speed up EM for GMM. They fall into two categories. The first category of methods divide data into segments (or overlapping canopies [11]) and traverse each segment incrementally [12]. The effectiveness depends on the data partition quality, which varies across data sets especially when data dimensions are high.

The second category of methods consider only the most influential data points to a GMM module when updating the parameters of that module, as illustrated by Figure 1. They differ in the criteria used to determine the most influential data points. Some are based on the probabilities for a point being generated by that GMM module [12], [13], some on the probability changes between two iterations [14]. We call them together *Elastic EM*, in the sense that they all allow for an elastic control of the amount data points considered for updating a GMM module.

Elastic EM avoids the data partition complexities faced by the first category, and is hence easier to use and shows promising results. It is the focus of this current work.

By allowing the consideration of only part of the dataset for a GMM module update, Elastic EM offers an easy way for programmers to trade quality of EM for speed. Although it shows some promise, in practice, the rate of speed improvement is often less substantial than the rate of quality loss. There are proposals of using conjugate gradient [15] and quasi-Newton method [16] to replace the M-step for performance, but they could cause pre-mature convergences of the algorithm. Effectively improving the speed of Elastic EM *without compromising the result quality* is the key for it to meet a broader range of usage.

This work attempts to achieve the goal by introducing

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00083

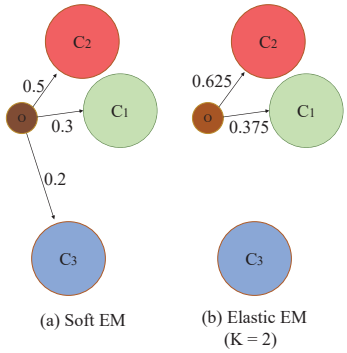

Figure 1. In Soft EM, each observation influences the updates of all mixture components. In Elastic EM, each observation influences only K mixture components that have the highest probabilities generating that observation (probabilities are normalized in (b))

a set of optimizations, which use bounds of probabilities to detect and avoid calculations in Elastic EM, improving the speed of Elastic EM without causing loss to the result quality. More specifically, through careful examination of Elastic EM, we have come up with a series of calculation filters, based on probability bounds obtained through Singular Value Decomposition and Triangle Inequality of Mahalanobis Distance. The interplay of these filters helps remove unnecessary calculations in the E-step of Elastic EM.

We further introduce a Delta optimization to also avoid many unnecessary calculations in the M-step of Elastic EM (for the setting $K = 1$). By putting these optimizations together, we create an improved Elastic EM, which we call *Lean Elastic EM* or LEEM. Experimental results on six diverse datasets show that LEEM on average removes 50% probability calculations and achieves up to 5X speedup with the popular setting.

We organize the rest of this paper as follows. In section II, we give a formal description of GMM and how Soft EM and Elastic EM are used to train GMMs. Section III describes the optimization approaches in detail. Section IV presents the experimental results and comparisons with Elastic EM and Soft EM. The last section summarizes the work.

II. Background

A. Gaussian Mixture Model

Definition 1. *Gaussian Mixture Model is a weighted sum of probability densities of its all mixture components:*

$$P(\mathbf{y_j}|\theta) = \sum_{m=1}^{M} \alpha_m \phi(\mathbf{y_j}|\mu_\mathbf{m}, \mathbf{\Sigma_m}) \tag{1}$$

where θ represents all parameters of the GMM, $\theta = \{\alpha_1, \alpha_2, \ldots, \alpha_M, \mu_\mathbf{1}, \mu_\mathbf{2}, \ldots, \mu_\mathbf{M}, \mathbf{\Sigma_1}, \mathbf{\Sigma_2}, \ldots, \mathbf{\Sigma_M}\}$. $\mathbf{y_j} = (y_{j,1}, y_{j,2}, \ldots, y_{j,d})$ is a d-dimensional continuous-valued vector. α_m is the weight of m^{th} mixture component, $m = 1, 2, \ldots M$. $\phi(\mathbf{y_j}|\mu_\mathbf{m}, \mathbf{\Sigma_m})$ is the probability density function of the m^{th} mixture component which follows a d-variate Gaussian distribution:

$$\phi(\mathbf{y_j}|\mu_\mathbf{m}, \mathbf{\Sigma_m}) = \frac{1}{\sqrt{(2\pi)^d |\mathbf{\Sigma_m}|}} exp\Big\{\frac{(\mathbf{y_j} - \mu_\mathbf{m})(\mathbf{y_j} - \mu_\mathbf{m})^T}{-2\mathbf{\Sigma_m}}\Big\} \tag{2}$$

where $\mu_\mathbf{m}$ and $\mathbf{\Sigma_m}$ are the mean and covariance matrix of the m^{th} mixture component.

There are three important configurations related to using a GMM Model for a problem. *GMM Model Order* is the number of mixture components in a GMM. During the training process of GMM, it's likely that some items in the covariance matrix become very small. These small values inflict singular covariance matrix and degrade performance of trained GMMs [1]. *Variance limiting* is usually applied to deal with this problem by setting those values to a minimum value σ^2_{min} (2.22×10^{-16} in our implementations). *Type of covariance matrix* hinges on the feature dependency. Sometimes, for datasets with independent feature dimensions, covariance matrices are made diagonal by setting non-diagonal elements in them zero. But for cases with dependent features, full matrix should be used [6]. Our explorations focus on the general case and use full covariance matrices in our implementations.

B. EM algorithm

Training GMM is to find the MLE of its parameters. The log likelihood of GMM can be written as:

$$\begin{aligned} L(\theta) &= \log P(Y|\theta) = \log\{P(\mathbf{y_1}|\theta)P(\mathbf{y_2}|\theta)...P(\mathbf{y_N}|\theta)\} \\ &= \sum_{i=1}^{N} \log P(\mathbf{y_i}|\theta) \end{aligned} \tag{3}$$

where Y is the set of all observations, $Y = \{\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_N}\}$ and θ is parameters of GMM.

Since there is no closed-form solutions to Formula 3, EM algorithm is used to approximate the solution. It approaches the solution via iterations of E-step and M-step. For the $(i+1)^{th}$ iteration:

E-step: Use the parameters from the previous iteration to get the expectation of log likelihood with respect to the conditional distribution of γ given Y and $\theta^{(i)}$, as in:

$$Q(\theta|\theta^{(i)}) = E_{\gamma|Y,\theta^{(i)}}[\log P(Y, \gamma|\theta)] \tag{4}$$

where γ is the latent variable which represents memberships of observations and $\theta^{(i)}$ is the estimate of parameters after the i^{th} iteration.

M-step: Calculate the θ maximizing Formula 4, that is:

$$\theta^{(i+1)} = \arg\max_{\theta}(Q(\theta|\theta^{(i)})) \tag{5}$$

After each iteration, $L(\theta)$ is calculated. EM algorithm repeats E-step and M-step until the difference between $L(\theta)$

from two consecutive iterations is less than the convergence threshold.

In practice, when using EM for GMM, the MLE for each parameter is computed as follows:

Membership: probability for an observation $\mathbf{y_j}$ to belong to the mixture component m

$$\hat{\gamma}_{j,m} = \frac{\hat{\alpha}_m \phi(\mathbf{y_j}|\hat{\mu}_\mathbf{m}, \hat{\mathbf{\Sigma}}_\mathbf{m})}{\sum_{t=1}^{M} \hat{\alpha}_t \phi(\mathbf{y_j}|\hat{\mu}_\mathbf{t}, \hat{\mathbf{\Sigma}}_\mathbf{t})} \tag{6}$$

Weight: probability of selecting the mixture component m

$$\hat{\alpha}_m = \frac{\sum_{j=1}^{N} \hat{\gamma}_{j,m}}{N} \tag{7}$$

Mean: mean of the mixture component m

$$\hat{\mu}_\mathbf{m} = \frac{\sum_{j=1}^{N} \hat{\gamma}_{j,m} \mathbf{y_j}}{\sum_{j=1}^{N} \hat{\gamma}_{j,m}} \tag{8}$$

Covariance Matrix: covariance matrix of the mixture component m

$$\hat{\mathbf{\Sigma}}_\mathbf{m} = \frac{\sum_{j=1}^{N} \hat{\gamma}_{j,m} (\mathbf{y_j} - \hat{\mu}_\mathbf{m})^\mathbf{T} (\mathbf{y_j} - \hat{\mu}_\mathbf{m})}{\sum_{j=1}^{N} \hat{\gamma}_{j,m}} \tag{9}$$

where $\mathbf{y_j}$ and $\hat{\mu}_\mathbf{m}$ are row vectors.

On each EM iteration, parameters $(\hat{\alpha}_m, \hat{\mu}_\mathbf{m}, \hat{\mathbf{\Sigma}}_\mathbf{m})$ got from the previous iteration are used to update $\hat{\gamma}_{j,m}$ at E-step, then the updated $\hat{\gamma}_{j,m}$ is used to update those parameters in M-step. This process repeats until convergence.

C. Elastic EM

Elastic EM is a family of variants of EM. They consider only the data points that are most influential to a GMM module when updating the parameters of that module. The different variants [12], [13] use different ways to define what data points are important to a GMM module. In this work, we take one of them as the focus in our development, but note that the proposed optimizations are applicable to other variants as well.

In the Elastic EM we focus on, in each iteration, every observation influences only the K mixture components with the highest probabilities generating the observation, where $1 \leqslant K < M$; M is the number of mixture components.

Elastic EM first calculates the probabilities for an observation to belong to each mixture component and then selects K mixture components with the highest probabilities. Let S_j be the set of indices of the K mixture components for the observation $\mathbf{y_j}$, we have:

$$S_j = \text{K-}\arg\max_m (\hat{\gamma}_{j,m}) \tag{10}$$

After finding the K mixture components, the weight for each mixture component is adjusted accordingly, thus:

$$\hat{\alpha}_{j,m} = \begin{cases} \dfrac{\hat{\alpha}_m}{\sum_{t \in S_j} \hat{\alpha}_t}, & m \in S_j \\ 0, & otherwise \end{cases} \tag{11}$$

Weights of the K mixture components with the highest probabilities are normalized so that their sum equals to 1. Weights of other mixture components are set to 0.

Formally, $\hat{\alpha}_{j,m}$ replaces $\hat{\alpha}_m$ in Formula 6. The adjusted $\hat{\gamma}'_{j,m}$ in Elastic EM is:

$$\hat{\gamma}'_{j,m} = \frac{\hat{\alpha}_{j,m} \phi(\mathbf{y_j}|\hat{\mu}_\mathbf{m}, \hat{\mathbf{\Sigma}}_\mathbf{m})}{\sum_{t=1}^{M} \hat{\alpha}_{j,t} \phi(\mathbf{y_j}|\hat{\mu}_\mathbf{t}, \hat{\mathbf{\Sigma}}_\mathbf{t})} \tag{12}$$

As for M-step, Elastic EM uses the same formulas as Soft EM does. The only difference is that $\hat{\gamma}_{j,m}$ is replaced by $\hat{\gamma}'_{j,m}$ in those formulas.

III. LEEM: Optimized Elastic EM

Elastic EM helps save some computations in the M-step of Soft EM as every observation is used by only K mixture components to update their parameters. However, as an approximation method, the value of K hinges on the requirements of specific applications. It is subject to a tradeoff between time savings and quality loss. The smaller K is, the more time saved, but also the more quality loss caused. LEEM is designed to magnify the time savings by Elastic EM without causing any extra quality loss.

LEEM accelerates both the E-step and the M-step of Elastic EM. Its optimization of the E-step of Elastic EM is based on the following basic idea: Since the goal of the step is to identify K mixture components with the highest probabilities for each observation, it's possible to filter out some unnecessary calculations via carefully designed bounds. Similar ideas have been implemented in K-Means [17], KNN [18], [19], and other machine learning algorithms. However, strategies used in those studies can not be applied to Elastic EM, because they are based on Euclidean distance while Elastic EM is based on probability as shown in Formula 6.

To see how the idea plays out on Elastic EM, Algorithm 1 outlines the E-step algorithm in Elastic EM. For each observation $\mathbf{y_j}$, all probabilities of it belonging to every mixture components need to be calculated. Thus, the number of probability calculations is $O(MN)$, where M is the number of mixture components and N is the number of observations. After having all probabilities, Elastic EM selects K mixture components with the highest probabilities. Then weights $\alpha_{j,m}$, weighted probability densities $P_{j,m}$ and probabilities $\gamma_{j,m}$ are updated accordingly.

It's easy to see that the first step of probability calculations incurs the majority of overhead. The optimization of LEEM for E-step is to reduce the number of probability calculations. To identify K mixture components with the highest probabilities, according to Formula 6, the numerator is the key factor, since the denominator is the same for everyone. As for the numerator, the probability density of the observation plays an essential role. Therefore, the strategy of LEEM is to construct bounds of unknown probability densities and use them to filter out unnecessary calculations.

For example, suppose that we would like to find the maximum probability density of observation $\mathbf{y_j}$ in all mixture components and the upper bound of an unknown probability density of observation $\mathbf{y_j}$ in some mixture component is already smaller than the current maximum. There would be no need to calculate the exact value of that probability density, since we know it will not be the maximum. This strategy can help us remove unnecessary calculations and generate significant speedups. The challenge is how to construct tight bounds of multi-variate Gaussian probability density efficiently, and how to effectively use these bounds. LEEM uses two levels of bounds respectively attained in SVD and through Triangle Inequality.

The optimization of LEEM for the M-step of Elastic EM is based on the following insight. Across two consecutive iterations, the set of observations relevant to a mixture model often has only small differences. Therefore, some calculations could be possibly saved if the results of the previous iteration can be reused when updating the module in this iteration, by incorporating only the changed parts. We call this optimization *Delta optimization*. This optimization is particularly useful for the setting of $K = 1$ of Elastic EM.

We next explain these optimizations in detail.

A. Accelerate E-step

As aforementioned, optimizations of E-step center around bounds of Mahalanobis distances. Formally, Mahalanobis distance is defined as follows:

Definition 2. *The Mahalanobis distance of an observation $\mathbf{y_j}$ from a mixture component with mean $\mu_\mathbf{m}$ and covariance matrix $\mathbf{\Sigma_m}$ is:*

$$d_M(\mathbf{y_j}, \mu_\mathbf{m}) = \sqrt{(\mathbf{y_j} - \mu_\mathbf{m})\mathbf{\Sigma_m^{-1}}(\mathbf{y_j} - \mu_\mathbf{m})^\mathbf{T}} \quad (13)$$

where $\mathbf{y_j}$ and $\mu_\mathbf{m}$ are row vectors.

Comparing Formula 13 with Formula 2, it's easy to see that the key component of multi-variate Guassian probability density is a Mahalanobis distance between the observation and the mean of the mixture component. To get bounds of multi-variate Gaussian probability density, we capitalize on bounds of such Mahalanobis distance.

Bounds based on singular value: As shown in Formula 13, to calculate Mahalanobis distance, we need to get the inverse matrix of covariance matrix. Since our strategy aims to support full covariance matrix rather than just diagonal matrix, getting inverse matrix of covariance matrix directly is often difficult, and Singular Value Decomposition (SVD) is usually used. We give a quick review of SVD as the knowledge is necessary for the rest of the discussion.

Definition 3. *Suppose $\mathbf{A}$ is a real $m \times n$ matrix, then there exists a factorization, called a singular value decomposition of $\mathbf{A}$, of the form:*

$$\mathbf{A} = \mathbf{UDV^T} \quad (14)$$

Elastic EM E-step
input : Y:dataset of all observations, size of Y is N; M: the number of mixture components; K: the number of mixture components with highest probabilities to be considered by each observation.
output: L:Log likelihood of current parameters
$L = 0$;
for $j \leftarrow 1$ **to** N **do**
  $L_j = 0$; //L_j is Log likelihood for observation $\mathbf{y_j}$;
  for $m \leftarrow 1$ **to** M **do**
    $P_{j,m} = \hat{\alpha}_m \phi(\mathbf{y_j}|\hat{\mu}_\mathbf{m}, \hat{\mathbf{\Sigma}}_\mathbf{m})$ //weighted probability density of $\mathbf{y_j}$ in mixture component m;
  end
  $sumP_j = \sum_{t=1}^{M} P_{j,t}$;
  for $m \leftarrow 1$ **to** M **do**
    $\hat{\gamma}_{j,m} = P_{j,m}/sumP_j$ //$\hat{\gamma}_{j,m}$ is the probability of observation $\mathbf{y_j}$ to belong to mixture component m;
  end
  Find out the K largest $\hat{\gamma}_{j,m}$, and put their mixture component indices into S_j;
  $S_j = \text{K-arg max}\,(\hat{\gamma}_{j,m})$;
  for $m \leftarrow 1$ **to** M **do**
    Update $\hat{\alpha}_{j,m}$ according to Formula 11
  end
  for $m \leftarrow 1$ **to** M **do**
    $P^{'}_{j,m} = \hat{\alpha}_{j,m}/\alpha_m * P_{j,m}$;
    $P^{'}_{j} += P^{'}_{j,m}$;
  end
  $sumP^{'}_{j} = \sum_{t=1}^{M} P^{'}_{j,t}$;
  for $m \leftarrow 1$ **to** M **do**
    $\hat{\gamma}^{'}_{j,m} = P^{'}_{j,m}/sumP^{'}_{j}$;
  end
  $L_j = \log P^{'}_{j}$;
  $L += L_j$;
end
return L;

Algorithm 1: E-step of Elastic EM for GMM

where $\mathbf{U}$ is a $m \times m$ unitary matrix, $\mathbf{D}$ is a $m \times n$ diagonal matrix, $\mathbf{V}$ is a $n \times n$ unitary matrix.

SVD has some appealing geometry properties. The three matices represent three operations on a vector, as shown in Figure 2. $\mathbf{U}$ is a rotation, $\mathbf{D}$ is scaling and $\mathbf{V}$ is another rotation. Thus, multiplying an observation with $\mathbf{A}$ is to actually rotate the vector, scale it and rotate it again. Based on this property, we know that both $\mathbf{U}$ and $\mathbf{V}$ will not change the length of the vector, only $\mathbf{D}$ does. LEEM capitalizes on this property which we will describe later. $\mathbf{D} = diag(\sigma_1, \sigma_2, ..., \sigma_n)$ and $\sigma_1 \geqslant \sigma_2 \geqslant ... \geqslant \sigma_n$. $\mathbf{D}$ is a diagonal matrix with singular values on its diagonal and they are ordered in decreasing order.

Formula 14 gives the general definition of SVD for any real-value $m \times n$ matrix. But in our case, the covariance matrix is a positive-semidefinite and symmetric matrix. In practical implementation of GMM, singular value less than a very small positive threshold will be replaced by that threshold value [20]. Hence, the covariance matrix is

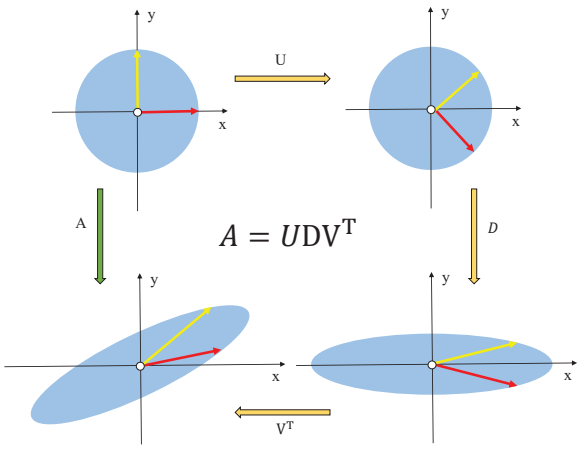

Figure 2. Geometrical transformations of SVD

positive-definite. For a positive-definite matrix, $\mathbf{U}$ and $\mathbf{V}$ are the same. Thus, Formula 14 can be rewritten as:

$$\mathbf{\Sigma} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathbf{T}} \tag{15}$$

After having decomposed matrices, inverse matrix of covariance matrix is easy to get, as in:

$$\mathbf{\Sigma}^{-1} = (\mathbf{U}\mathbf{D}\mathbf{U}^{\mathbf{T}})^{-1} = \mathbf{U}\mathbf{D}^{-1}\mathbf{U}^{\mathbf{T}} \tag{16}$$

where $\mathbf{D}^{-1} = diag(1/\sigma_1, 1/\sigma_2, ..., 1/\sigma_n)$.

Lemma 1. *The Mahalanobis distance of an observation $\mathbf{y_j}$ from a mixture component with mean $\mu_\mathbf{m}$ and covariance matrix $\mathbf{\Sigma_m}$ satisfies:*

$$\frac{|\mathbf{y_j} - \mu_\mathbf{m}|}{\sqrt{\sigma_1}} \leqslant d_M(\mathbf{y_j}, \mu_\mathbf{m}) \leqslant \frac{|\mathbf{y_j} - \mu_\mathbf{m}|}{\sqrt{\sigma_d}} \tag{17}$$

where $|\mathbf{y_j} - \mu_\mathbf{m}|$ is the Euclidean distance between $\mathbf{y_j}$ and $\mu_\mathbf{m}$, σ_1 and σ_d are the maximum and minimum singular value of $\mathbf{\Sigma_m}$ (d is the dimension of $\mathbf{y_j}$ and $\mu_\mathbf{m}$).

Proof: According to Formula 13, we have:

$$d_M^2(\mathbf{y_j}, \mu_\mathbf{m}) = (\mathbf{y_j} - \mu_\mathbf{m})\mathbf{\Sigma_m^{-1}}(\mathbf{y_j} - \mu_\mathbf{m})^\mathbf{T}$$

Plugging Formula 16 into it:

$$d_M^2(\mathbf{y_j}, \mu_\mathbf{m}) = (\mathbf{y_j} - \mu_\mathbf{m})\mathbf{U_m}\mathbf{D_m^{-1}}\mathbf{U_m^T}(\mathbf{y_j} - \mu_\mathbf{m})^\mathbf{T}$$

Let's assume: $\mathbf{y_j} - \mu_\mathbf{m} = (x_1, x_2, ..., x_d)$, $(\mathbf{y_j} - \mu_\mathbf{m})\mathbf{U_m} = (y_1, y_2, ..., y_d)$ and $\mathbf{D_m^{-1}} = diag(1/\sigma_1, 1/\sigma_2, ..., 1/\sigma_d)$, the Mahalanobis distance can be written as:

$$d_M^2(\mathbf{y_j}, \mu_\mathbf{m}) = \frac{y_1^2}{\sigma_1} + \frac{y_2^2}{\sigma_2} + ... + \frac{y_d^2}{\sigma_d}$$

We know $\sigma_1 \geqslant \sigma_2 \geqslant ... \geqslant \sigma_d$, hence we can get:

$$\frac{y_1^2 + y_2^2 + ... + y_d^2}{\sigma_1} \leqslant d_M^2(\mathbf{y_j}, \mu_\mathbf{m}) \leqslant \frac{y_1^2 + y_2^2 + ... + y_d^2}{\sigma_d}$$

Since $\mathbf{U_m}$ will not change the length of vectors:

$$|(\mathbf{y_j} - \mu_\mathbf{m})\mathbf{U_m}| = |\mathbf{y_j} - \mu_\mathbf{m}|$$

Plugging it into previous inequality, we have:

$$\frac{|\mathbf{y_j} - \mu_\mathbf{m}|^2}{\sigma_1} \leqslant d_M^2(\mathbf{y_j}, \mu_\mathbf{m}) \leqslant \frac{|\mathbf{y_j} - \mu_\mathbf{m}|^2}{\sigma_d}$$

Since $d_M^2(\mathbf{y_j}, \mu_\mathbf{m})$, σ_1 and σ_d are all positive, we thereby have Formula 17 by getting square roots of them. ■

The benefits of using these bounds is that to get the bound, we only need to calculate Euclidean distance between the observation $\mathbf{y_j}$ and the mean of the mixture component $\mu_\mathbf{m}$. If the bound works, it can help us filter out the calculation of many Mahalanobis distances which are much more computationally expensive.

Bounds based on triangle inequality:

Theorem 1. *Triangle inequality holds in Mahalanobis distance.*

Recall that *triangle inequality* says that the sum of the lengths of any two sides of a triangle must be greater than or equal to the length of the remaining side. In the context of Mahalanobis distance, let $d_M(\mathbf{x}, \mathbf{y})$ represent the Mahalanobis distance between two points $\mathbf{x}$ and $\mathbf{y}$. Theorem 1 says that for any three points $\mathbf{a}, \mathbf{b}, \mathbf{c}$, we have:

$$|d_M(\mathbf{a}, \mathbf{b}) - d_M(\mathbf{b}, \mathbf{c})| \leqslant d_M(\mathbf{a}, \mathbf{c}) \leqslant d_M(\mathbf{a}, \mathbf{b}) + d_M(\mathbf{b}, \mathbf{c}) \tag{18}$$

Proof: Let **a - b = v**, **c - b = w**, **a - c = u** and $\mathbf{D^{-1}} = \mathbf{S}\mathbf{S^T}$

Since $\mathbf{D^{-1}} = diag(1/\sigma_1, 1/\sigma_2, ..., 1/\sigma_n)$, we have $\mathbf{S} = diag(1/\sqrt{\sigma_1}, 1/\sqrt{\sigma_2}, ..., 1/\sqrt{\sigma_n})$.

We have already shown in Lemma 1 proof:

$$d_M(\mathbf{a}, \mathbf{b}) = \sqrt{(\mathbf{a} - \mathbf{b})\mathbf{U}\mathbf{D^{-1}}\mathbf{U^T}(\mathbf{a} - \mathbf{b})^\mathbf{T}}$$

Let $\mathbf{E} = \mathbf{U}\mathbf{S}$, then:

$$d_M(\mathbf{a}, \mathbf{b}) = \sqrt{\mathbf{v}\mathbf{E}\mathbf{E}^\mathbf{T}\mathbf{v}^\mathbf{T}}$$

Let $\mathbf{v}\mathbf{E} = \mathbf{v}'$, $\mathbf{w}\mathbf{E} = \mathbf{w}'$, $\mathbf{u}\mathbf{E} = \mathbf{u}'$, we have:

$$d_M(\mathbf{a}, \mathbf{b}) = \sqrt{\mathbf{v}'\mathbf{v}'^\mathbf{T}} = |\mathbf{v}'|$$

Similarly, $d_M(\mathbf{b}, \mathbf{c}) = |\mathbf{w}'|$, $d_M(\mathbf{a}, \mathbf{c}) = |\mathbf{u}'|$

$$\begin{aligned} d_M(\mathbf{a}, \mathbf{c}) = |\mathbf{u}'| = |\mathbf{u}\mathbf{E}| = |(\mathbf{v} - \mathbf{w})\mathbf{E}| \\ = |\mathbf{v}\mathbf{E} - \mathbf{w}\mathbf{E}| = |\mathbf{v}' - \mathbf{w}'| \end{aligned}$$

According to triangle inequality for Euclidean distance, we know:

$$||\mathbf{v}'| - |\mathbf{w}'|| \leqslant |\mathbf{v}' - \mathbf{w}'| \leqslant |\mathbf{v}'| + |\mathbf{w}'|$$

Therefore,

$$|d_M(\mathbf{a}, \mathbf{b}) - d_M(\mathbf{b}, \mathbf{c})| \leqslant d_M(\mathbf{a}, \mathbf{c}) \leqslant d_M(\mathbf{a}, \mathbf{b}) + d_M(\mathbf{b}, \mathbf{c})$$

■

Corollary 1. *Assume $\mu_{\mathbf{m}}, \mu_{\mathbf{s}}, \mathbf{y_j}$ form a triangle, then the Mahalanobis distance of the observation $\mathbf{y_j}$ from the mixture component with mean $\mu_{\mathbf{m}}$ and covariance matrix $\mathbf{\Sigma_m}$ satisfies:*

$$d_M(\mathbf{y_j}, \mu_{\mathbf{m}}) \geqslant \begin{cases} D_{m,s} - \frac{|\mathbf{y_j} - \mu_{\mathbf{s}}|}{\sqrt{\sigma_d}}, \ (D_{m,s} \geqslant \frac{|\mathbf{y_j} - \mu_{\mathbf{s}}|}{\sqrt{\sigma_d}}) \\ \frac{|\mathbf{y_j} - \mu_{\mathbf{s}}|}{\sqrt{\sigma_1}} - D_{m,s}, \ (D_{m,s} \leqslant \frac{|\mathbf{y_j} - \mu_{\mathbf{s}}|}{\sqrt{\sigma_1}}) \end{cases} \tag{19}$$

$$d_M(\mathbf{y_j}, \mu_{\mathbf{m}}) \leqslant \frac{|\mathbf{y_j} - \mu_{\mathbf{s}}|}{\sqrt{\sigma_d}} + D_{m,s} \tag{20}$$

where
$D_{m,s} = d_M(\mu_{\mathbf{s}}, \mu_{\mathbf{m}}) = \sqrt{(\mu_{\mathbf{s}} - \mu_{\mathbf{m}})\mathbf{\Sigma_m^{-1}}(\mu_{\mathbf{s}} - \mu_{\mathbf{m}})^{\mathbf{T}}}$, *$\sigma_1$ and σ_d are the maximum and minimum singular value of $\mathbf{\Sigma_m}$.*

Proof: First, note that all the Mahalanobis distances in Corollary 1 are with $\mathbf{\Sigma_m}$.

According to Theorem 1:

$$|d_M(\mathbf{y_j}, \mu_{\mathbf{s}}) - D_{m,s}| \leqslant d_M(\mathbf{y_j}, \mu_{\mathbf{m}}) \leqslant d_M(\mathbf{y_j}, \mu_{\mathbf{s}}) + D_{m,s}$$

According to Lemma 1, we have:

$$\frac{|\mathbf{y_j} - \mu_{\mathbf{s}}|}{\sqrt{\sigma_1}} \leqslant d_M(\mathbf{y_j}, \mu_{\mathbf{s}}) \leqslant \frac{|\mathbf{y_j} - \mu_{\mathbf{s}}|}{\sqrt{\sigma_d}}$$

Therefore, the upper bound is easy to get:

$$d_M(\mathbf{y_j}, \mu_{\mathbf{m}}) \leqslant d_M(\mathbf{y_j}, \mu_{\mathbf{s}}) + D_{m,s} \leqslant \frac{|\mathbf{y_j} - \mu_{\mathbf{s}}|}{\sqrt{\sigma_d}} + D_{m,s}$$

As for the lower bound, there are three cases:

(1) $D_{m,s} \geqslant |\mathbf{y_j} - \mu_{\mathbf{s}}|/\sqrt{\sigma_d}$

$$\begin{aligned} d_M(\mathbf{y_j}, \mu_{\mathbf{m}}) &\geqslant |d_M(\mathbf{y_j}, \mu_{\mathbf{s}}) - D_{m,s}| = D_{m,s} - d_M(\mathbf{y_j}, \mu_{\mathbf{s}}) \\ &\geqslant D_{m,s} - \frac{|\mathbf{y_j} - \mu_{\mathbf{s}}|}{\sqrt{\sigma_d}} \end{aligned}$$

(2) $D_{m,s} \leqslant |\mathbf{y_j} - \mu_{\mathbf{s}}|/\sqrt{\sigma_1}$

$$\begin{aligned} d_M(\mathbf{y_j}, \mu_{\mathbf{m}}) &\geqslant |d_M(\mathbf{y_j}, \mu_{\mathbf{s}}) - D_{m,s}| = d_M(\mathbf{y_j}, \mu_{\mathbf{s}}) - D_{m,s} \\ &\geqslant \frac{|\mathbf{y_j} - \mu_{\mathbf{s}}|}{\sqrt{\sigma_1}} - D_{m,s} \end{aligned}$$

(3) $|\mathbf{y_j} - \mu_{\mathbf{s}}|/\sqrt{\sigma_1} < D_{m,s} < |\mathbf{y_j} - \mu_{\mathbf{s}}|/\sqrt{\sigma_d}$

The sign of $d_M(\mathbf{y_j}, \mu_{\mathbf{s}}) - D_{m,s}$ is unknown and it's possible $d_M(\mathbf{y_j}, \mu_{\mathbf{s}}) - D_{m,s} = 0$. Therefore, the lower bound is 0 in this case, which is not helpful. ■

To use the bounds, $D_{m,s}$ needs to be calculated. Therefore, the number of introduced Mahalanobis distance calculation is $O(M^2)$ (M is the number of mixture components.) Since M is usually a small number, the overhead is small. In practice, when we calculate the probability for an observation $\mathbf{y_j}$ to belong to the m^{th} mixture component, LEEM leverages all previous μ_s $(s = 1, 2, ...m-1)$ to construct bounds according to Corollary 1. As Figure 3 illustrates, as the distances to previous μ_s are computed earlier already, LEEM just picks the tightest one among all those bounds without the need to recompute them.

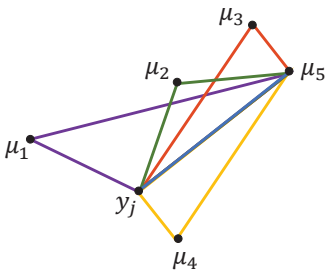

Figure 3. $\mu_1, \mu_2, \mu_3, \mu_4$ are used to construct bounds for $d_M(\mathbf{y_j}, \mu_5)$. Since $|\mathbf{y_j} - \mu_{\mathbf{s}}|$, $(s = 1, 2, 3, 4)$ are already known, no need to recompute these bounds; LEEM just picks the tightest one.

Filtering conditions: The filtering condition used by LEEM can be described as follows:

> Let $UB(\hat{\gamma}'_{j,m})$ be the upper bound of $\hat{\gamma}'_{j,m}$ and γ_j^K be the K^{th} largest one of all probabilities that have been already calculated for observation $\mathbf{y_j}$. If $UB(\hat{\gamma}'_{j,m}) \leqslant \gamma_j^K$, then we can skip the calculation of $\hat{\gamma}'_{j,m}$.

This is because we know $\hat{\gamma}'_{j,m}$ will not be one of the K highest probabilities for $\mathbf{y_j}$ and $\mathbf{y_j}$ will not be assigned to that mixture component.

Lemma 1 and Corollary 1 have given the bounds of the Mahalanobis distance of the observation $\mathbf{y_j}$ from the mixture component with mean $\mu_{\mathbf{m}}$ and covariance matrix $\mathbf{\Sigma_m}$. But we need the upper bound of $\hat{\gamma}'_{j,m}$ in the filtering condition. It's easy to get this upper bound via the lower bound of the Mahalanobis distance, as in:

$$UB(\hat{\gamma}'_{j,m}) = \frac{\hat{\alpha}_{j,m} UB(\phi(\mathbf{y_j}|\hat{\mu}_{\mathbf{m}}, \hat{\mathbf{\Sigma}}_{\mathbf{m}}))}{\sum_{t=1}^{M} \hat{\alpha}_{j,t} \phi(\mathbf{y_j}|\hat{\mu}_{\mathbf{t}}, \hat{\mathbf{\Sigma}}_{\mathbf{t}})} \tag{21}$$

$$\begin{aligned} &UB(\phi(\mathbf{y_j}|\hat{\mu}_{\mathbf{m}}, \hat{\mathbf{\Sigma}}_{\mathbf{m}})) \\ &= \frac{1}{\sqrt{(2\pi)^d |\hat{\mathbf{\Sigma}}_{\mathbf{m}}|}} exp\Big\{ \frac{[LB(d_M(\mathbf{y_j}, \hat{\mu}_{\mathbf{m}}))]^2}{-2} \Big\} \end{aligned} \tag{22}$$

where UB and LB stand for upper bound and lower bound.

The bounds are straightforward to understand. We need the upper bound of the probability and there is a "-2" in the exponential term; it hence corresponds to the lower bound of the Mahalanobis distance. The lower bound is always positive.

In practice, for an observation $\mathbf{y_j}$, LEEM maintains a list of its current K highest probabilities and their corresponding mixture components' indices. Every time when the upper bound of an unknown probability has been calculated, that upper bound is compared with the lowest probability in that list. If the upper bound is lower than the lowest probability, we can filter out the calculation for that unknown probability. Since there are two ways to calculate the upper bound, if

one upper bound fails, LEEM tries the other. If no upper bound works, the probability calculation can not be removed and the exact value is calculated. After getting the value, LEEM updates the list with the exact value. The list is materialized through the use of *min-heap*. Min-heap has some properties that make it a desirable choice. On min-heap, finding the lowest probability takes $O(1)$ time while inserting and deleting a new probability take $O(\log(K))$ time, where K is the size of the min-heap.

B. Accelerate M-step for K=1

Besides the E-step optimization, LEEM incorporates a Delta optimization into the M-step. This optimization is only for the case where $K = 1$. This case is also called *hard EM* [21]. It is one of the most popular settings of Elastic EM that has been used in applications [22]–[24]. So even though this optimization is for this setting only, it is meaningful for practical usage of Elastic EM.

For the case of $K = 1$, the formula in M-Step for calculating covariance matrix after t iterations is:

$$\Sigma_m^{(t)}(j,k) = \frac{1}{|C_m^{(t)}|}\sum_{x_i \in C_m^t}(x_{i,j} - \bar{x}_j^{(t)})(x_{i,k} - \bar{x}_k^{(t)}) \quad (23)$$

where the superscript t indicates the number of iterations. Thus, $\Sigma_m^{(t)}(j,k)$ means the element at the j^{th} row and k^{th} column of the covariance matrix of the m^{th} mixture component after t iterations. $C_m^{(t)}$ is a set of all observations assigned to the m^{th} mixture component after t iterations. $|C_m^{(t)}|$ is the size of the set. $\bar{x}_j^{(t)}$ and $\bar{x}_k^{(t)}$ are the j^{th} and k^{th} dimension of the mean of the m^{th} mixture component after t iterations.

If we consider the scalar multiplication as a measure, which is $|C_m^{(t)}| + 1$ using Formula 23.

We derived another way to calculate covariance matrix which leverages the results in the previous iteration. The number of calculations tapers off with the process of convergence. It can be shown as follows:

$$\begin{aligned}\Sigma_m^{(t+1)}(j,k) = {} & \frac{1}{|C_m^{(t+1)}|}(|C_m^{(t)}|\Sigma_m^{(t)}(j,k) + |C_m^{(t)}|\Delta\bar{x}_j\Delta\bar{x}_k) \\ & - \frac{1}{|C_m^{(t+1)}|}\sum_{x_i \in A}(x_{i,j} - \bar{x}_j^{(t+1)})(x_{i,k} - \bar{x}_k^{(t+1)}) \\ & + \frac{1}{|C_m^{(t+1)}|}\sum_{x_i \in B}(x_{i,j} - \bar{x}_j^{(t+1)})(x_{i,k} - \bar{x}_k^{(t+1)}) \quad (24)\end{aligned}$$

where $\Delta\bar{x}_j = \bar{x}_j^{(t+1)} - \bar{x}_j^{(t)}$ and $\Delta\bar{x}_k = \bar{x}_k^{(t+1)} - \bar{x}_k^{(t)}$, A is the set of observations that leave the m^{th} mixture component in the $(t+1)^{th}$ iteration, B is the set of observations that join the m^{th} mixture component in the $(t+1)^{th}$ iteration. Formally, $C_m^{(t+1)} = C_m^{(t)} - A + B$.

Proof: We now prove the correctness of Formula 24.

According to Formula 23:

$$\Sigma_m^{(t+1)}(j,k) = \frac{1}{|C_m^{(t+1)}|}\sum_{x_i \in C_m^{(t+1)}}(x_{i,j} - \bar{x}_j^{(t+1)})(x_{i,k} - \bar{x}_k^{(t+1)})$$

Since $C_m^{(t+1)} = C_m^{(t)} - A + B$, plug it into previous equation:

$$\begin{aligned}\Sigma_m^{(t+1)}(j,k) = {} & \frac{1}{|C_m^{(t+1)}|}\sum_{x_i \in C_m^{(t)}}(x_{i,j} - \bar{x}_j^{(t+1)})(x_{i,k} - \bar{x}_k^{(t+1)}) \\ & - \frac{1}{|C_m^{(t+1)}|}\sum_{x_i \in A}(x_{i,j} - \bar{x}_j^{(t+1)})(x_{i,k} - \bar{x}_k^{(t+1)}) \\ & + \frac{1}{|C_m^{(t+1)}|}\sum_{x_i \in B}(x_{i,j} - \bar{x}_j^{(t+1)})(x_{i,k} - \bar{x}_k^{(t+1)}) \quad (25)\end{aligned}$$

Because $\bar{x}_j^{(t+1)} = \bar{x}_j^{(t)} + \Delta\bar{x}_j$ and $\bar{x}_k^{(t+1)} = \bar{x}_k^{(t)} + \Delta\bar{x}_k$,

$$\begin{aligned}& \sum_{x_i \in C_m^{(t)}}(x_{i,j} - \bar{x}_j^{(t+1)})(x_{i,k} - \bar{x}_k^{(t+1)}) \\ = {} & \sum_{x_i \in C_m^{(t)}}(x_{i,j} - \bar{x}_j^{(t)})(x_{i,k} - \bar{x}_k^{(t)}) - \sum_{x_i \in C_m^{(t)}}\Delta\bar{x}_j(x_{i,k} - \bar{x}_k^{(t)}) \\ & - \sum_{x_i \in C_m^{(t)}}\Delta\bar{x}_k(x_{i,j} - \bar{x}_j^{(t)}) + \sum_{x_i \in C_m^{(t)}}\Delta\bar{x}_j\Delta\bar{x}_k \quad (26)\end{aligned}$$

Because:

$$\sum_{x_i \in C_m^{(t)}}\Delta\bar{x}_j(x_{i,k} - \bar{x}_k^{(t)}) = \Delta\bar{x}_j\sum_{x_i \in C_m^{(t)}}(x_{i,k} - \bar{x}_k^{(t)}) = 0$$

$$\sum_{x_i \in C_m^{(t)}}\Delta\bar{x}_k(x_{i,j} - \bar{x}_j^{(t)}) = \Delta\bar{x}_k\sum_{x_i \in C_m^{(t)}}(x_{i,j} - \bar{x}_j^{(t)}) = 0$$

Plug them into Formula 26, we have:

$$\begin{aligned}& \sum_{x_i \in C_m^{(t)}}(x_{i,j} - \bar{x}_j^{(t+1)})(x_{i,k} - \bar{x}_k^{(t+1)}) \\ = {} & \sum_{x_i \in C_m^{(t)}}(x_{i,j} - \bar{x}_j^{(t)})(x_{i,k} - \bar{x}_k^{(t)}) + \sum_{x_i \in C_m^{(t)}}\Delta\bar{x}_j\Delta\bar{x}_k \\ = {} & |C_m^{(t)}|\Sigma_m^{(t)}(j,k) + |C_m^{(t)}|\Delta\bar{x}_j\Delta\bar{x}_k \quad (27)\end{aligned}$$

Plugging Formula 27 into Formula 25, we get Formula 24. ■

The benefits of using Formula 24 over 23 are on the reuse of the previous covariance matrix. Instead of recalculating the covariance matrix, LEEM only calculates changes brought by observations which change their belonging mixture components during consecutive iterations. Those observations are often just a small portion of all observations. It hence avoids substantial computational overhead. $\Sigma_m^{(t)}(j,k)$ is already known. When we consider the number of scalar multiplications, Formula 24 takes $(4 + |A| + 1 + |B| + 1)$ scalar multiplications, where $|A|$ and $|B|$ are the size of these two sets. Therefore, the speedup is approximately $\frac{|C_m^{(t)}|+1}{|A|+|B|+6}$. When the LEEM approaches convergence, $|A|$ and $|B|$ are usually very small. Therefore, the speedup can be significant.

Table I
DATASETS USED IN EXPERIMENTS

Dataset	Dataset Properties			
	# sample	# dim	# category	avg category size
Dota2	102944	116	2	51472
Skin Seg.	245057	4	2	122529
Pen Digits	10992	16	10	1099
Drive Diagno.	58509	49	11	5319
Credit Card	30000	24	2	15000
Cover Type	581012	54	7	83002

IV. EXPERIMENTS

We evaluate the efficacy of LEEM on a variety of benchmark datasets from UCI Machine Learning Repository [25] as shown in Table I. Skin Segmentation [26] contains skin textures from images of diversity of age, gender, and race people. Dataset of Pen digits [27] collects coordinate information of hand-written digits by 44 writers. In Drive diagnosis dataset [28], samples are the extracted drive signals from 12 different operating conditions. Default of credit card dataset [29] is composed of basic personal information (e.g. gender, education, marital status, etc) and history of past payment, which aims to predict the probability of default. Cover type dataset [30] contains cartographic data related to forest cover type. Dota2 [25] is a dataset used for predicting the winner of game by evaluating the heroes chosen by players of each team. Every dataset has multiple categories and each category is modeled as a GMM.

Our experiments measure the efficacy of LEEM based on the number of probability calculations it removes as well as the speedup it attains. All of our experiments are conducted in the following experimental environment: PowerEdge R620 equipped with 2 Xeon E5-2670 CPUs, 128GB memory, Red Hat Enterprise Linux 7.5.

Experiments are conducted on Soft EM, Elastic EM and LEEM. All three algorithms are implemented based on OpenCV 3.4.1 [20]. Soft EM is already implemented in OpenCV, we hence directly use that implementation. Elastic EM and LEEM[1] are both implemented by modifying the source code of the Soft EM so that we can keep the comparison fair by avoiding the influence from other factors (e.g. different libraries, different optimization levels, etc).

The GMM Model order (i.e., the number of mixture models) needs to be set when applying GMM. There is no universal approach to choosing the optimal model order. Akaike information criterion(AIC) [31] and Bayesian information criterion (BIC) [32] are widely used to estimate the optimal model order. In our experiments, we use AIC to select the model order from a series of predefined values. Regarding variance limiting, we use the same value OpenCV uses, which is about 2.22×10^{-16}. Full covariance matrix is used in our experiments.

[1]Code of LEEM is available at https://github.com/PICTureRG/LEEM

A. Avoided Calculations

Table II reports the number of probability calculations of Elastic EM and LEEM for each dataset with different Ks. The number of LEEM is the total number of probability calculations including both the ones left and extra calculations introduced by our optimizations. The percentages shown in the parentheses are the ratios of the numbers of calculations in LEEM to those in Elastic EM.

It's easy to see that LEEM avoids the largest number of probability calculations when $K = 1$. For most of the datasets, LEEM is able to remove 50% probability calculations. When K gets larger, the savings become smaller. This is intuitive. When $K = 1$, only the mixture component with the highest probability is relevant. Every time the upper bound of an unknown probability is compared with the current maximum, the filtering condition has a large chance to be met. But when K equals to a large number, the K^{th} largest number could be small, and the filtering condition is harder to get satisfied. In practical usage of Elastic EM, K is usually small.

In our experiments, the best performance is achieved on Skin segmentation dataset. It removes 93% probability calculations when $K = 1$. Overall, the results in Table I show that the dataset with a larger category size tends to get more savings in probability calculations. A larger category entails more computations in EM, and also usually takes more iterations to converge, giving more room for improvement.

B. Speedups

In Table III, we report the running time of each part of algorithms and the speedups that LEEM brings. The numbers in the parenthesis indicate speedups over Soft EM. The running time we report includes all extra overhead introduced by our optimizations.

For the E-step, the Elastic EM does all calculations that Soft EM does without optimizations; therefore, their running times are almost the same as those of Soft EM. LEEM achieves on average 2.3X speedups when $K = 1$, which is consistent with the calculation savings shown in Table II. The speedups of Elastic EM are from M-Step. Our optimization methods for M-step has brought LEEM significant speedups when $K = 1$. All datasets get speedups of one or two orders of magnitude. LEEM attains 101.7X speedups on Skin segmentation dataset at M-step with $K = 1$, thanks to the large category size of the dataset. As mentioned in last section, the speedup of M-step is approximately $\frac{|C_m^{(t)}|+1}{|A|+|B|+6}$. A larger category size usually leads to larger $|C_m^{(t)}|$. As this optimization applies only to $K = 1$, the speedups of M-step at other K values are similar to those in Elastic EM.

Table II
PROBABILITY CALCULATIONS IN DIFFERENT ALGORITHMS

Dataset	M	K	No. of Probability Calculations*	
			Elastic EM	LEEM
Dota2	12	1	839525	451303 (54%)
		2	905752	609558 (67%)
		4	893985	738956 (83%)
		8	900497	859885 (95%)
Skin Seg.	20	1	2684594	191650 (7%)
		2	2605875	426788 (16%)
		5	2398880	939782 (39%)
		10	2691421	1920246 (71%)
Pen Digits	5	1	24615	14140 (57%)
		2	22368	20959 (94%)
		3	22707	22616 (99.6%)
		4	23717	23683 (99.8%)
Drive Diag.	15	1	524279	413898 (79%)
		2	532952	469610 (88%)
		4	532952	489428 (92%)
		8	528838	495876 (94%)
Credit Card	12	1	444153	219756 (49%)
		2	435377	312798 (72%)
		4	435377	384240 (88%)
		8	421310	416440 (99%)
Cover Type	20	1	10289174	5948359 (58%)
		2	9363640	7531113 (80%)
		5	11460390	9769904 (85%)
		10	10992196	10287371 (94%)

M: The number of mixture components.
K: The number of mixture components with highest probabilities to be considered by each observation.
$*$: The number is the total number of probability calculations divided by the total number of iterations of all GMMs in that dataset. A percentage in parentheses is the number of probability calculations in LEEM over those in Elastic EM.

The overall speedups of LEEM vary across the different settings. When $K = 1$, the improvement is the largest. The best performance is on dataset of Skin segmentation, which achieves about 5X speedup over Elastic EM and 11.5X speedup over Soft EM. When K gets larger, the improvements become less significant. Since in practice, the value of K is usually a small number, LEEM is an appealing replacement of the current Elastic EM.

V. CONCLUSION

This study revisits Elastic EM and proposes LEEM as a practical replacement of Elastic EM. LEEM applies two types of bounds based on the singular value and triangle inequality to detect unnecessary probability calculations. A Delta optimization for M-Step is also incorporated into LEEM. LEEM on average achieves 5X speedups over Soft EM. It speeds up Elastic EM by up to 2.3X with no quality loss, appearing as a promising drop-in replacement of current Elastic EM.

Acknowledgement This material is based upon work supported by DOE Early Career Award (DE-SC0013700), the National Science Foundation (NSF) under Grant No. CCF-1455404, CCF-1525609, CNS-1717425, CCF-1703487. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DOE or NSF.

REFERENCES

[1] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE transactions on speech and audio processing*, vol. 3, no. 1, pp. 72–83, 1995.

[2] M. A. Zissman, "Automatic language identification using gaussian mixture and hidden markov models," in *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, vol. 2. IEEE, 1993, pp. 399–402.

[3] S. G. Koolagudi, A. Barthwal, S. Devliyal, and K. S. Rao, "Real life emotion classification using spectral features and gaussian mixture models," *Procedia engineering*, vol. 38, pp. 3892–3899, 2012.

[4] P. A. Torres-Carrasquillo, T. P. Gleason, and D. A. Reynolds, "Dialect identification using gaussian mixture models," in *ODYSSEY04-The Speaker and Language Recognition Workshop*, 2004.

[5] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2. IEEE, 2004, pp. 28–31.

[6] H. Permuter, J. Francos, and I. Jermyn, "A study of gaussian mixture models of color and texture features for image classification and segmentation," *Pattern Recognition*, vol. 39, no. 4, pp. 695–706, 2006.

[7] Z. Ji, Y. Xia, Q. Sun, Q. Chen, D. Xia, and D. D. Feng, "Fuzzy local gaussian mixture model for brain mr image segmentation," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 3, pp. 339–347, 2012.

[8] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[9] A. Senior, G. Heigold, M. Bacchiani, and H. Liao, "Gmm-free dnn acoustic model training," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5602–5606.

[10] 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)*, pp. 1–38, 1977.

[11] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 169–178.

[12] R. M. Neal and G. E. Hinton, "A view of the em algorithm that justifies incremental, sparse, and other variants," in *Learning in graphical models*. Springer, 1998, pp. 355–368.

[13] B. Thiesson, C. Meek, and D. Heckerman, "Accelerating em for large databases," *Machine Learning*, vol. 45, no. 3, pp. 279–299, 2001.

[14] F.-X. Jollois and M. Nadif, "Speed-up for the expectation-maximization algorithm for clustering categorical data," *Journal of Global Optimization*, vol. 37, no. 4, pp. 513–525, 2007.

[15] M. Jamshidian and R. I. Jennrich, "Conjugate gradient acceleration of the em algorithm," *Journal of the American Statistical Association*, vol. 88, no. 421, pp. 221–228, 1993.

[16] K. Lange, "A quasi-newton acceleration of the em algorithm," *Statistica sinica*, pp. 1–18, 1995.

Table III
RUNNING TIME COMPARISON OF DIFFERENT ALGORITHMS

Dataset	M	K	Running time per iteration (ms) *								
			Total			E-Step			M-Step		
			Soft EM	Elastic EM	LEEM	Soft EM	Elastic EM	LEEM	Soft EM	Elastic EM	LEEM
Dota2	12	1	15597	6433 (2.4)	3474 (4.5)	4642	4733 (1.0)	2773 (1.7)	10841	1583 (6.8)	579 (19.0)
		2		7027 (2.2)	5643 (2.8)		4635 (1.0)	3302 (1.4)		2291 (4.7)	2243(4.8)
		4		8897 (1.8)	8008 (1.9)		4748 (1.0)	4024 (1.2)		4040 (2.7)	3871 (2.8)
		8		12166 (1.3)	11503 (1.4)		4892 (1.0)	4688 (1.0)		7145 (1.5)	6700 (1.6)
Skin Seg.	20	1	7366	3086 (2.4)	641 (11.5)	1726	1825 (1.0)	552(3.1)	5595	1127 (4.6)	55 (101.7)
		2		3093 (2.4)	2032 (3.6)		1747 (1.0)	689 (2.5)		1302 (4.3)	1300 (4.3)
		5		3243 (2.3)	2639 (2.8)		1540 (1.1)	910 (1.9)		1662 (3.4)	1687 (3.3)
		10		5004 (1.5)	4789 (1.5)		1859 (0.9)	1608 (1.1)		3098 (1.8)	3133 (1.8)
Pen Digits	5	1	17.8	12.2 (1.5)	7.3 (2.4)	4.6	6.1 (0.8)	5.3 (0.9)	12.6	5.2 (2.4)	1.1 (12.1)
		2		11.6 (1.5)	13 (1.4)		4.9 (0.9)	5.9 (0.8)		5.9 (2.1)	6.4 (2.0)
		3		14.1 (1.3)	15.8 (1.1)		5.3 (0.9)	6.4 (0.7)		8.0 (1.6)	8.8 (1.4)
		4		17.2 (1.0)	15.9 (1.1)		6.1 (0.8)	6.0 (0.8)		10.4 (1.2)	9.3 (1.4)
Drive Diag.	15	1	648	301 (2.2)	221 (2.9)	187	194 (1.0)	169 (1.1)	446	92 (4.8)	37 (12.1)
		2		329 (2.0)	318 (2.0))		198 (0.9)	190 (1.0)		115 (3.9)	113 (3.9)
		4		408 (1.6)	384 (1.7)		215 (0.9)	204 (0.9)		176 (2.5)	164 (2.7)
		8		517 (1.3)	510 (1.3)		217 (0.9)	215 (0.9)		283 (1.6)	279 (1.6)
Credit Card	15	1	988	463 (2.1)	246 (4.0)	257	303 (0.9)	220 (1.2)	725	152 (4.8)	18 (40.3)
		2		492 (2.0)	486 (2.0)		293 (0.9)	271 (0.9)		192 (3.8)	207 (3.5)
		4		570 (1.7)	573 (1.7)		293 (0.9)	303 (0.8)		271 (2.7)	264 (2.7)
		8		709 (1.4)	738 (1.3)		285 (0.9)	317 (0.8)		418 (1.7)	414 (1.8)
Cover Type	20	1	14669	5970 (2.5)	3358 (4.4)	4775	4589 (1.0)	2907 (1.6)	9692	1145 (8.5)	205 (47.3)
		2		5840 (2.5)	5267 (2.8)		4280 (1.1)	3694 (1.3)		1345 (7.2)	1353 (7.2)
		5		7489 (2.0)	6815 (2.2)		5188 (0.9)	4715 (1.0)		2092 (4.6)	1905 (5.1)
		10		7569 (1.9)	7568 (1.9)		5040 (0.9)	5129 (1.0)		2327 (4.2)	2246 (4.3)

M: The number of mixture components.
K: The number of mixture components with largest probabilities to be considered by each observation.
$*$: the numbers in parentheses are speedups (X) over Soft EM

[17] Y. Ding, Y. Zhao, X. Shen, M. Musuvathi, and T. Mytkowicz, "Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup," in *International Conference on Machine Learning*, 2015, pp. 579–587.

[18] G. Chen, Y. Ding, and X. Shen, "Sweet knn: An efficient knn on gpu through reconciliation between redundancy removal and regularity," in *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 2017, pp. 621–632.

[19] S. Yang and X. Shen, "Falcon: A fast drop-in replacement of citation knn for multiple instance learning," in *Proceedings of the 2018 ACM on Conference on Information and Knowledge Management*. ACM, 2017.

[20] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[21] G. Celeux and G. Govaert, "A classification em algorithm for clustering and two stochastic versions," *Computational statistics & Data analysis*, vol. 14, no. 3, pp. 315–332, 1992.

[22] A. Samé, C. Ambroise, and G. Govaert, "A classification em algorithm for binned data," *Computational statistics & data analysis*, vol. 51, no. 2, pp. 466–480, 2006.

[23] A. Samé , C. Ambroise, and G. Govaert, "An online classification em algorithm based on the mixture model," *Statistics and Computing*, vol. 17, no. 3, pp. 209–218, 2007.

[24] K.-K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 1, pp. 39–51, 1998.

[25] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[26] R. B. Bhatt, G. Sharma, A. Dhall, and S. Chaudhury, "Efficient skin region segmentation using low complexity fuzzy decision tree model," in *India Conference (INDICON), 2009 Annual IEEE*. IEEE, 2009, pp. 1–4.

[27] F. Alimoglu, E. Alpaydin, and Y. Denizhan, "Combining multiple classifiers for pen-based handwritten digit recognition," 1996.

[28] F. Paschke, C. Bayer, M. Bator, U. Mönks, A. Dicks, O. Enge-Rosenblatt, and V. Lohweg, "Sensorlose zustandsüberwachung an synchronmotoren," in *Proceedings. 23. Workshop Computational Intelligence, Dortmund, 5*, 2013, p. 211.

[29] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2473–2480, 2009.

[30] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and electronics in agriculture*, vol. 24, no. 3, pp. 131–151, 1999.

[31] H. Akaike, "A new look at the statistical model identification," *IEEE transactions on automatic control*, vol. 19, no. 6, pp. 716–723, 1974.

[32] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.