Approximating the Covariance Matrix of GMMs with Low-rank Perturbations

Malik Magdon-Ismail

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
Email: magdon@cs.rpi.edu

Jonathan T. Purnell

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
Email: purnej@cs.rpi.edu

Abstract:

Covariance matrices capture correlations that are invaluable in modeling real-life datasets. Using all d^2 elements of the covariance (in d dimensions) is costly and could result in over-fitting; and the simple diagonal approximation can be over-restrictive. In this work, we present a new model, the Low-Rank Gaussian Mixture Model (LRGMM), for modeling data which can be extended to identifying partitions or overlapping clusters. The curse of dimensionality that arises in calculating the covariance matrices of the GMM is countered by using low-rank perturbed diagonal matrices. The efficiency is comparable to the diagonal approximation, yet one can capture correlations among the dimensions. Our experiments reveal the LRGMM to be an efficient and highly applicable tool for working with large high-dimensional datasets.

Keywords: Gaussian Mixture models; efficient; maximum likelihood; E-M

Biographical notes:

Jonathan Purnell received his PhD in Computer Science at Rensselaer Polytechnic Institute, Troy, NY. He is currently in a post-doctoral position under the US Army grant for the Social/Cognitive Network Academic Research Center (SCNARC) at Rensselaer Polytechnic Institute. His research interests include machine learning theory, clustering algorithms and modeling probabilistic distributions.

1 Introduction

Fitting models to a dataset or clustering samples based on some metric are useful tools in better understanding internal structures (e.g. communities) or identifying the underlying distribution of the data, like Ramdane et al. (2010) and Wan et al. (2009). With the rise of Internet-based social networking, there are increasingly larger and larger datasets to process. Although the runtime of clustering and modeling algorithms can be decreased by dimension reduction like Garg and Murty (2009), the advantage may be outweighed by the loss of information. Since the sample covariance matrices provide important information about the probability distribution of the samples, we address the high statistical and computational complexity of the covariance matrix introduced by the quadratic dependence on d, the data dimension. An additional drawback of the full covariance matrix in high dimension is that the additional $O(d^2)$ parameters can lead to heavy overfitting. In order to address these difficulties, we propose a new matrix decomposition.

Due to the computational cost associated with the full covariance matrix, an approximation of the covariance matrix is often used instead. The problem then becomes one of balancing computational cost with accuracy. A simple approximation is to use the diagonal of the covariance matrix or, in other words, use only the dimension by dimension variances. As we explore further in Section 3, there are several other proposed approximations which mostly utilize decompositions of the covariance matrix, but which are not suitable for GMMs.

The LRGMM model preserves most of the structure of the full sample-based covariance matrix while maintaining a training time that is linear in d. Thus, it is computationally comparable to the diagonal approximation of the covariance matrix while yielding a more accurate model. We also demonstrate that the LRGMM does not succumb as much to over fitting the data. Our method is tested against the diagonal approximation and the full covariance matrix for Gaussian Mixture Models. The accuracy and runtime performances of the LRGMM are then compared to using full covariance matrices and diagonal matrix approximations. This comparison is performed on both synthetic and real-world data.

2 Gaussian Mixture Model

The Gaussian Mixture Model (GMM) is ubiquitous both in general and within computer science. It has been applied to a variety of problems ranging across topics such as clustering, image segmentation, computational finance, speech recognition, and biometrics. It is also the foundation of our work presented in this paper, so we briefly review this model. Formally, a mixture model is a linear combination of component densities $p(\mathbf{x}|i)$ and is defined by

$$p(\mathbf{x}) = \sum_{i=1}^{M} p(\mathbf{x}|i)\pi_i. \tag{1}$$

For the GMM, the component densities are Gaussian (Normal) densities, where component i is specified by its center, μ_i, and variance, σ_i^2.

Given a set of N data samples, $\mathbf{x}_n, n = 1 \ldots N$, the goal is to use them to calculate the parameters of our GMM. The likelihood of a GMM is defined by $\mathcal{L} = \prod_{n=1}^{N} p(\mathbf{x}_n)$. We can write this into an error function to be minimized,

$$E = -\ln \mathcal{L} = -\sum_{n=1}^{N} \ln p(\mathbf{x}_n) = -\sum_{n=1}^{N} \ln \left\{ \sum_{i=1}^{M} p(\mathbf{x}_n|i)\pi_i \right\} . \tag{2}$$

In most cases, including this paper, this error function is minimized by the Expectation-Maximization (EM) algorithm. The details for this algorithm are well detailed by Bishop (1995). Also, extending these equations to multiple dimensions is fairly straightforward and also well known.

Random initial parameters often lead to local optimum and require several runs before finding an acceptable result. A more common solution for initial parameters is to use K-means clustering. With K-means clustering we can quickly find initial clusters and their centers. The covariance matrices and mixture weights can also be calculated from each cluster. An additional measure is to use K-means with $M \ln M$ random initial centers and ignore the $\ln M$ smallest clusters as in Dasgupta and Schulman (2000). While these approaches will not eliminate the problem of local minima, which is a byproduct of the error function, in practice they will greatly reduce the chance that the local minima found will be far from the true distribution of the data. We note that there are algorithms that can detect an optimal number of clusters, such as Jing et al. (2009), but we have chosen our method to maintain a linear runtime.

3 Covariance Matrix Approximation

3.1 Problem Definition

The covariance is the defining characteristic of the GMM. Typically the EM algorithm is used for training a GMM on a given dataset. The expectation step uses the inverse of the covariance matrix to calculate the probability of each sample and the maximization step updates the covariance matrix using these probabilities. The runtime of a single step of this EM algorithm is $O(NMd^2)$, where we have N data samples and M mixtures in the model. This can be prohibitive for high dimension problems, and thus one often uses an approximation to the full covariance. The ideal approximation for the full covariance is one that is not only accurate and calculated quickly, but also has an inverse that can be efficiently used to calculate sample probabilities quickly.

Since we are focusing on the GMM, our metric is the log likelihood.

$$\ln \mathbf{L} = -\frac{1}{2} \sum_{i=1}^{N} (x_i - \mu)^T \hat{\Sigma}^{-1} (x_i - \mu) + \frac{N}{2} \ln |\hat{\Sigma}^{-1}| - \frac{Nd}{2} \ln 2\pi \ , \tag{3}$$

where N is the number of samples, d is the sample dimension, and $\hat{\Sigma}$ is the estimate of Σ. Our discussion will focus on a single component. All our arguments extend to mixtures with multiple components. After differentiating, the log likelihood is maximized for

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \Sigma = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)(x_i-\mu)^T \ . \tag{4}$$

This can be computed in $O(Nd^2)$. The basic analytic task we address is how to efficiently choose $\hat{\Sigma}$, under a sparsity constraint, so as to do so efficiently.

3.2 Previous Work

There are many methods of matrix approximation, each having their advantages and drawbacks. One common approach to matrix approximation is by decomposition. In Pourahmadi (2007), the author notes the difficulty of estimating positive definite matrices and uses decompositions to address the problem. The Cholesky decomposition is shown as a better decomposition than variance-correlation or spectral decomposition. However, the time complexity of the Cholesky decomposition is quadratic with respect to the dimension of the matrix.

There has been some work in covariance matrix approximations. These involve a variety of approaches, mainly decomposition like El Karoui (2008) and Pourahmadi (2007), statistical estimation like Haan and Levin (1997), and using assumptions on the matrix such as Chaudhuri et al. (2007) and Astrand et al. (2007). These previous approaches each suffer from one of two drawbacks: a time complexity that grows quadratically with the data dimension or inability to be efficiently inverted.

Our Contribution: We have proposed a new matrix decomposition to approximate the covariance matrices of a Gaussian Mixture Model. We use a low-rank perturbation of a diagonal matrix and approximate the inverse of the covariance matrix. The conjugate gradient method is used to optimize the parameters with respect to the log probability formula of a GMM. By exploiting the structure of the covariance matrix and approximating its inverse, we develop an algorithm that is bounded linearly with respect to sample dimension, making it comparable to the diagonal approximation.

4 Approximating the Full Covariance

4.1 Diagonal Approximation

First we will cover the diagonal approximation. We rewrite our log likelihood equation with the diagonal matrix D, where $\Delta\mathbf{x}_t = \mathbf{x}_n - \mu$ and $\hat{\Sigma} = D \approx \Sigma = \frac{1}{N}\sum_n^N \Delta\mathbf{x}_n \Delta\mathbf{x}_n^T$

$$\ln \mathbf{L} = -\frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n-\mu)^T D^{-1}(\mathbf{x}_n-\mu) + \frac{N}{2}\ln|D^{-1}| - \frac{Nd}{2}\ln 2\pi \tag{5}$$

which is maximized when we minimize

$$\varepsilon = \frac{1}{N}\sum_{n=1}^{N} \Delta\mathbf{x}_n^T D^{-1}\Delta\mathbf{x}_n - \ln|D^{-1}| \tag{6}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{d} \frac{\Delta\mathbf{x}_n(i)^2}{D_{ii}} - \sum_{i=1}^{d} \ln\frac{1}{D_{ii}}. \tag{7}$$

We can now solve analytically for D by setting the derivative to zero

$$\frac{\partial\varepsilon}{\partial D_{ii}} = \frac{1}{N}\sum_{n=1}^{N}\Delta\mathbf{x}_n(i)^2 - D_{ii} = 0, D_{ii} = \frac{1}{N}\sum_{n=1}^{N}\Delta\mathbf{x}_n(i)^2.$$

The derivation is given in Section (5). With the diagonal constraint, the maximizer of $\ln\mathbf{L}$ is the diagonal of Σ. This can be computed in $O(Nd)$, but loses all off-diagonal correlations. This loss can be seen in Figure 1, where we compare models resulting from a full covariance matrix with those using a diagonal matrix

4.2 Optimal Rank-k Decomposition

Another popular decomposition is the eigenvalue decomposition $\Sigma = \sum_{i=1}^{k}\lambda_i\mathbf{u}_i\mathbf{u}_i^T$, where λ_i and $\mathbf{u}_i$ are the i^{th} eigenvalue and eigenvector of Σ. This is the optimal rank-k decomposition under any unitarily invariant norm. However, this is not an invertible decomposition and so it is not appropriate for GMMs which use Σ^{-1}.

5 Low-Rank Perturbed Diagonal Decomposition (LRPDD)

Our approximation to Σ is using a low-rank perturbation of a diagonal matrix

$$\Sigma^{-1} \approx D^2 + \mathbf{a}\mathbf{a}^T \tag{8}$$

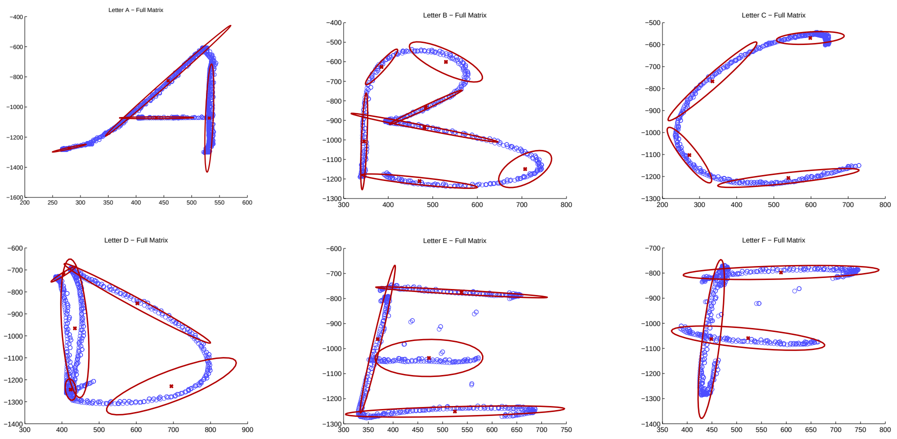
where D is our diagonal matrix and $\mathbf{a}$ is a d-dimensional vector that defines the low-rank perturbation. We use D^2 to ensure the approximation is positive semi-definite without having to add constraints

The log-likelihood of a GMM, using this approximation is

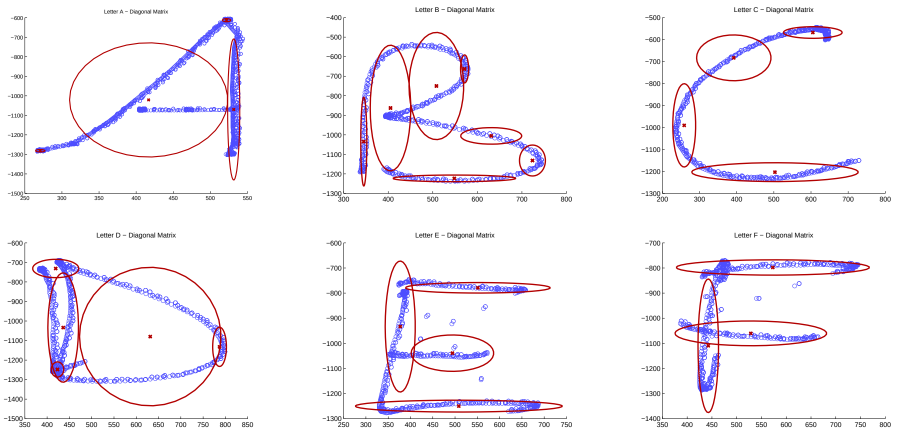
$$\ln\mathbf{L} = -\frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n-\mu)^T(D^2+\mathbf{a}\mathbf{a}^T)(\mathbf{x}_n-\mu) + \frac{N}{2}\ln|(D^2+\mathbf{a}\mathbf{a}^T)| - \frac{Nd}{2}\ln 2\pi \tag{9}$$

where N is the number of samples and d is the sample dimension. Our goal is to maximize Eqn. (9). Thus, our optimization problem is to minimize

$$\varepsilon = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\mu)^T(D^2+\mathbf{a}\mathbf{a}^T)(\mathbf{x}_n-\mu) - \ln|(D^2+\mathbf{a}\mathbf{a}^T)| \tag{10}$$



Full Covariance Matrix



Diagonal Covariance Matrix

Figure 1 Alphabet data fitted by GMMs with Full and Diagonal Matrix

where D is a diagonal matrix with diagonal elements D_{ii}, and $\mathbf{a}$ is a vector $[a_1, \ldots, a_d]^T$. We may also formulate ε as

$$\varepsilon = ||\Sigma^{-1} - (D^2 + \mathbf{a}\mathbf{a}^T)||_F^2 \tag{11}$$

for a given Σ, where $||x||_F$ is the Frobenius norm. These two problems define a new matrix decomposition which preserves rank.

Intuitively, it may seem that using the diagonal of Σ for D is sufficient. However, this requires fitting $\mathbf{a}\mathbf{a}^T$ to a Σ with zeros along the diagonal. This pulls $\mathbf{a}$ away from its optimal value and towards a saddle-point at $\mathbf{a} = 0^d$. In finding the optimal values for our parameters, D and $\mathbf{a}$, we search for the point where the gradient of our optimization equation reaches zero. First, we find the gradient with respect to the α element of D, denoted D_α:

$$\frac{\partial \varepsilon}{\partial D_\alpha} = \sum_{n=1}^{N} \mathbf{x}_{n\alpha}^T D \mathbf{x}_{n\alpha} - 2D_\alpha^{-1} + \frac{\text{trace}(2D^\alpha D^{-2}\mathbf{a}\mathbf{a}^T D^{-2})}{1 + \mathbf{a}^T D^{-2}\mathbf{a}} \tag{12}$$

where the derivative of the $\ln |(D^2 + \mathbf{a}\mathbf{a}^T)|$ term is derived using Jacobi's formula for the differential of a matrix determinant, $d\det(A) = \text{trace}(\text{adj}(A)dA)$; D^α is the matrix D with all elements set to zero except $D_\alpha\alpha$; and trace() is the trace function. This formula can be written in vector form as:

$$\frac{\partial \varepsilon}{\partial D} = 2diag(\Sigma)D - 2D^{-1} + \frac{2D(D^{-2}\mathbf{a})^2}{1 + \mathbf{a}^T D^{-2}\mathbf{a}} \tag{13}$$

The gradient for $\mathbf{a}$ can be derived in a similar manner

$$\frac{\partial \varepsilon}{\partial \mathbf{a}_\alpha} = \frac{2}{N}\sum_{n=1}^{N}\left(\sum_{i=1}^{d} x_n(i)\mathbf{a}(i)\right)^2 x_{n\alpha} - \text{trace}\left((D^2 + \mathbf{a}\mathbf{a}^T)^{-1}\frac{\partial(D^2 + \mathbf{a}\mathbf{a}^T)}{\partial \mathbf{a}_\alpha}\right) \tag{14}$$

$$= \frac{2}{N}\sum_{n=1}^{N}\left(\sum_{i=1}^{d} x_n(d)\mathbf{a}_(d)\right)^2 x_{n\alpha} - \frac{2\mathbf{a}_\alpha}{D_\alpha} + \frac{2\mathbf{a}_\alpha}{D_\alpha}\sum_{i=1}^{d}\frac{\mathbf{a}_(d)^2}{D_{ii}} \tag{15}$$

which can be put into vector form as

$$\frac{\partial \varepsilon}{\partial a} = 2\Sigma\mathbf{a} - 2D^{-2}\mathbf{a} + \frac{2(D^{-2}\mathbf{a})(D^{-2}\mathbf{a})^T\mathbf{a}}{1 + \mathbf{a}^T D^{-2}\mathbf{a}} \tag{16}$$

The solution of these equations is non-trivial. We first looked at setting to zero the gradient with respect to $\mathbf{a}$. Through some simplification we obtain from Eqn. (16) the following three equivalent conditions for $\mathbf{a}$:

$$\Sigma\mathbf{a} = 2\frac{D^{-2}\mathbf{a}}{1 + \mathbf{a}^T D^{-2}\mathbf{a}} \tag{17}$$

$$2(1 + \mathbf{a}^T D^{-2}\mathbf{a})D^2\Sigma\mathbf{a} = \mathbf{a} \tag{18}$$

$$\mathbf{a} = \frac{(D^2\Sigma)^{-1}\mathbf{a}}{1 + \mathbf{a}^T D^{-2}\mathbf{a}}. \tag{19}$$

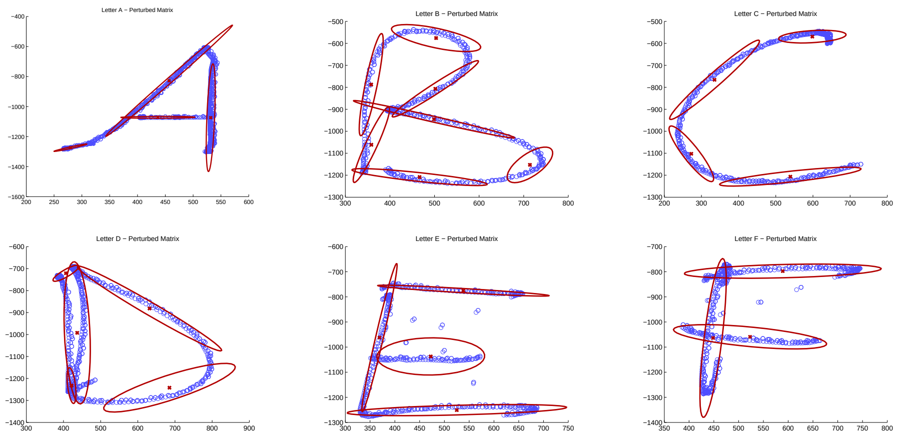
Each of these equations give us some information. Eqn. (17) shows that this is a generalized eigenvalue problem. Despite the symmetric positive semi-definite nature of the covariance matrix, there does not seem to be a linear-time solution to this problem. Eqn. (18) is a potential candidate for updating a in an iterative manner (where the left-hand side values are from initial values or the previous iteration). Unfortunately, multiplying by Σ makes this an expanding equation and divergent from the optimal value of a. Eqn. (19) gives us a contracting version of Eqn. (18), but requires multiplying by the inverse of Σ. Multiplying by Σ can be done in $O(Nd)$, but multiplying by Σ^{-1} cannot.

We also looked at setting to zero the gradient with respect to D. From Eqn. (13), we can simplify down to the following equations:

$$\frac{D^{-2}}{1 + \mathbf{a}^T D^{-2}\mathbf{a}} = \text{diag}(\Sigma) \tag{20}$$

$$D^{-2} = \text{diag}(\Sigma)(1 + \mathbf{a}^T D^{-2}\mathbf{a}) \tag{21}$$

$$D^{-2} = \text{diag}(\Sigma) + \frac{(D^{-2}\mathbf{a})^2}{1 + \mathbf{a}^T D^{-2}\mathbf{a}}. \tag{22}$$



LRPDD Matrix

Figure 2 Alphabet data fitted by GMMs with LRPDD

Again, although these equations are coupled with the value of $\mathbf{a}$, they provide some insight into D. First, Eqn. (20) supports the intuition that the optimal value of D involves a contribution from $\mathbf{a}$. Second, as with the gradient for $\mathbf{a}$, Eqn. (21) and Eqn. (22) give us two possible equations to compliment Eqn. (18) and Eqn. (19) in an iterative update approach. However, since the problems of multiplying by Σ and Σ^{-1} still remain for $\mathbf{a}$, the efficiency of these equations for D do not lead to an overall efficient approach to maximizing the parameters.

By using an approximation based on a low-rank perturbation of a diagonal matrix, our intention is to obtain an improved accuracy over the diagonal approximation but maintain a linear bound with respect to the sample dimension. The low-rank perturbation adds another d parameters over the diagonal approximations d parameters. Therefore we expect the computational cost to be higher than the diagonal approximation, but still well below that of the full-rank approximation which has $d(d+1)/2$ parameters. Further, by directly approximating the inverse of the covariance matrix, the cost of inversion is avoided. Note that this low rank perturbation implies a similar low rank perturbation expression for Σ itself. The difference between the LRPDD and the diagonal matrix approximation can be seen in Figure 2, comparing to Figure 1.

5.1 Iterative Solution Via Conjugate Gradient Optimization

Instead of an analytical solution, we use the conjugate gradient descent from Hestenes and Stiefel (1952) to reach an optimum (See Algorithm 1 for a summary of the conjugate gradient algorithm). We initialize $\mathbf{a}$ to be a random vector, whose elements are random samples from the uniform distribution in $[0, 1]$; D is initialized as the inverse of the diagonal of Σ, which can be easily calculated in linear time. The gradient referred to as g in the algorithm is just the concatenation of the gradient for $\mathbf{a}$ and the gradient for D. The search direction is referred to as v. The threshold, τ_g, is set to 0.001, our stopping criterion. Although one generally

Algorithm 1 Conjugate Gradient

```
1: Input: data $x$, diagonal matrix $D_0$, perturbation $\mathbf{a}_0$, threshold $\tau_g$
2: $g_0 \leftarrow [\frac{\partial \varepsilon}{\partial \mathbf{a}_0}, \frac{\partial \varepsilon}{\partial D_0}]$ ; $v_0 \leftarrow -g_0$.
3: while $|g_i| > \tau_g$ do
4:     Perform a line search along $v_i$ to find optimal step size, $s$.
5:     $\mathbf{a}_{i+1} \leftarrow \mathbf{a}_i + s \cdot v_i(1 \ldots d)$.
6:     $D_{i+1} \leftarrow D_i + s \cdot v_i(d+1 \ldots 2d)$.
7:     $g_{i+1} \leftarrow [\frac{\partial \varepsilon}{\partial a_{i+1}}, \frac{\partial \varepsilon}{\partial D_{i+1}}]$
8:     $\beta \leftarrow \frac{g_{i+1}(g_{i+1}^T g_i)}{g_i^T g_i}$
9:     Calculate new direction, $v_i \leftarrow -g_{i+1} + \beta v_i$.
10:    Update gradient, $g_i \leftarrow g_{i+1}$
11: end while
```

expects to use d steps of the conjugate gradient method, practice shows that only a few steps are needed.

With these formulas and algorithms, we can perform the EM algorithm for the LRGMM efficiently in $O(Nd)$ time. For the expectation step, we calculate the log likelihoods of each data point for each component, using Eqn. (9). First, we calculate the log-determinant of the covariance matrix

$$\ln |(D^2 + \mathbf{a}\mathbf{a}^T)| = \ln(\mathbf{a}^T D^{-1} \mathbf{a} + 1) + \ln |D|. \tag{23}$$

Calculating $|D|$ and $\mathbf{a}^T D^{-1}\mathbf{a}$ is straightforward and only requires one pass over the parameters. We add to this term the log of the prior term, $\ln \pi$. Finally, we compute the remainder of Eqn. (9)

$$\begin{aligned} &\frac{1}{N}\sum_{n=1}^{N} \Delta\mathbf{x}_n (D^2 + \mathbf{a}\mathbf{a}^T) \Delta\mathbf{x}_n \\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{d} \Delta\mathbf{x}_n^2(i) D_i^2 + \frac{1}{N}\sum_{n=1}^{N}(\sum_{i=1}^{d} \Delta\mathbf{x}_n^2(i)\mathbf{a}_i)^2. \end{aligned} \tag{24}$$

Computing this term of Eqn. (9) only requires $O(Nd)$ time and, thus, computing the log-likelihood only takes $O(MNd)$ time.

The equations for the maximization step, Eqn. (13) and Eqn. (16), have been derived. Since D is a diagonal matrix, multiplying D or D^{-1} by a vector takes only $O(d)$ time. For Eqn. (13), we need to multiply by the diagonal of Σ. This can be done in efficiently, since

$$\Sigma_{ii} = \sum_{n=1}^{N} (\mathbf{x}_n(i) - \mu_n(i))^2. \tag{25}$$

A difficult problem arises when we compute Eqn. (16). We notice the need to solve $\Sigma\mathbf{a}$ efficiently. Again, by using the definition of Σ, we can solve this efficiently;

$$\Sigma\mathbf{a} = (\sum_{n=1}^{N} (\mathbf{x}_n - \mu)(\mathbf{x}_n - \mu)^T)\mathbf{a} = \sum_{n=1}^{N} (\mathbf{x}_n - \mu)(\mathbf{x}_n^T\mathbf{a} - \mu^T\mathbf{a}) \tag{26}$$

$$= \sum_{n=1}^{N} (\mathbf{x}_n - \mu)\beta_n = \sum_{n=1}^{N} \mathbf{x}_n^T \beta_n - \mu \sum_{n=1}^{N} \beta_n, \tag{27}$$

where $\beta_n = \mathbf{x}_n^T \mathbf{a} - \mu^T \mathbf{a}$. This can be solved in four steps, each only requiring $O(Nd)$ time. First, we compute $\mu^T \mathbf{a}$. Second, we compute β_n for $n = 1, \ldots, N$. Each value of β_n only needs to compute $\mathbf{x}_n^T \mathbf{a}$, so computing all of them can be done in $O(Nd)$ time. Thirdly, given β_n, we compute $\sum_{n=1}^{N} \mathbf{x}_n^T \beta_n$ and $C = \sum n = 1^N \beta_n$ simultaneously, which again can be done in $O(Nd)$ time. Finally, we finish the equation by subtracting μC from the result of previous step. Thus, having an efficient solution for $\Sigma \mathbf{a}$, we have an efficient solution for computing the gradients for both D and $\mathbf{a}$. Further, this means that the conjugate gradient approach can be efficiently executed. As we stated earlier, in practice the number of conjugate gradient iterations needed for accurate approximations is much less than the data dimension or number of data samples. So, it can be expected that the LRGMM can be accurately trained in nearly linear time.

So far, we have discussed LRPDD for a single mixture. In order to extend to multiple mixtures, we need to introduce mixture weights, π_k. These values determine the contribution of each mixture to the entire model. Mixture weights can also be viewed as prior probabilities that determine the chance a data point was generated by a particular mixture. Many of the details of these parameters were reviewed in the discussion on the Gaussian Mixture Model in Section 2. The main concern of training multiple mixtures is calculating the posterior probabilities of the data samples with respect to each model, p_{nk}. These values are calculated using Eqn. (9) with the added term $\ln \pi_k$, to account for the mixture weight. The posterior probabilities are then normalized for each data sample over all mixtures; $\sum_{k=1}^{M} p_{nk} = 1$. For the maximization step, the mixture weights are approximated by the posterior probabilities; $\pi_k = \frac{1}{N} \sum_{n=1}^{N} p_{nk}$. Also, in estimating the values for D and $\mathbf{a}$ for mixture k, data samples $\mathbf{x}_n$ are weighted by the respective posterior probability, p_{tk}. The mixture means are now updated by $\mu_k = \frac{1}{T} \sum_{n=1}^{N} p_{tk} \mathbf{x}_n$. When calculating the gradients for D and $\mathbf{a}$, an element on the diagonal of Σ and β_n now become $\Sigma_{ii} = \sum_{n=1}^{N} p_{tk} (\mathbf{x}_n(i) - \mu_k(i))^2$ and $\beta_n = p_{tk}(\mathbf{x}_n^T \mathbf{a}_k - \mu_k^T \mathbf{a}_k)$.

In the next section, we explore how our approximation compares to the diagonal approximation and the full rank covariance matrix.

6 Experiments

Synthetic Data

A key property of the LRPDD is that it captures the correlation information of the covariance matrix. To visualize the capabilities of the LRPDD, we compare the LRPDD and diagonal approximations with the full matrix in Figure 3.

To qualitatively illustrate the performance of our algorithm, we synthesized GMM data. Training datasets consisted of samples generated by a GMM with six clusters, randomly generated means, and randomly generated positive semi-definite covariance matrices for each mixture. An experiment consists of training a GMM on a dataset using one of the approximation methods described above

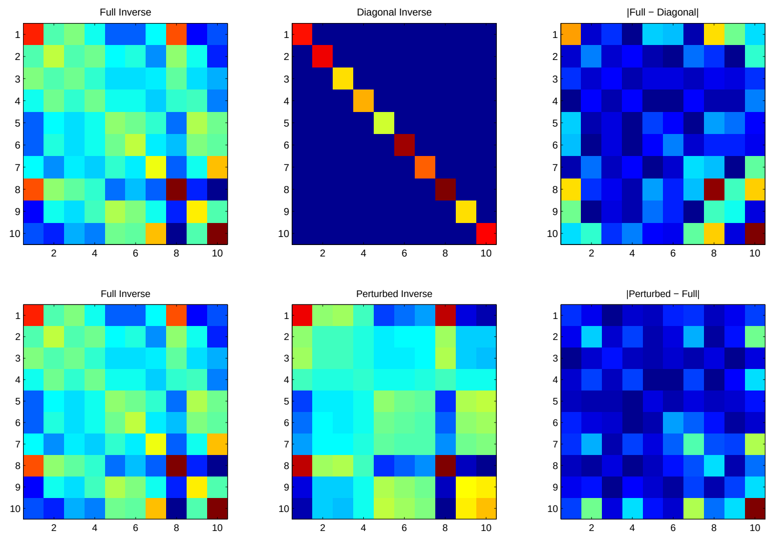


Figure 3 Differences in approximating a matrix with the LRPDD and Diagonal Methods

Dims.	Diag.	Pert.	Full
10	-109	-101	-87
50	-1032	-1005	-734
100	-2250	-2219	-1615

Table 1 In-sample average log probabilities for N=10k,M=6 (div. by 1000)

Dims.	Diag.	Pert.	Full
10	-109	-101	-88
50	-948	-925	-868
100	-2243	-2189	-2050

Table 2 In-sample average log probabilities for N=1k,M=6 (div. by 100)

(full covariance matrix, diagonal, or low rank perturbed). The results from our experiments with these datasets are shown in Tables 1 - 4.

6.0.1 Training Performance

We show performance from two perspectives, the in-sample and out-sample average log probabilities. Experiments were performed over various sample dimensions, dataset sizes, and number of mixtures. Tables 1 to 4 show results for datasets of different sample size, N, and number of mixtures, M. 100 datasets were generated for each pair of sample dimension and dataset size. The average log probabilities are calculated over the log probabilities from the 100 datasets. The values have been divided by 1000 to make comparison easier.

From these tables, we can see a clear trend that the perturbed matrices provide a better approximation over the diagonal matrices. Since we see this in both our

Dims.	Diag.	Pert.	Full
10	-111	-103	-100
50	-1035	-1009	-897
100	-2254	-2199	-1994

Table 3 In-sample average log probabilities for N=10k,M=3 (div. by 1000)

Dims.	Diag.	Pert.	Full
10	-11	-10.2	-8.95
50	-95.6	-94.5	-95.6
100	**-225.9**	**-232.2**	**-249.1**

Table 4 Out-sample average log probabilities N=1k,M=6 (div. by 1000)

Dims.	Pert	Full	Full/Pert
50	18.04	45.65	2.53
100	33.39	162.92	4.88

Table 5 Runtimes using the Full matrix and the Perturbed approximation with T10k

in-sample and out-sample accuracies, we can be assured that the performance of the perturbed matrix is not due to overfitting. Further, the out-sample accuracies bolded in Table 4, show that, at high dimensions, there is a greater potential for the full matrix to overfit than the perturbed matrix. The LRPDD provides the user a compromise between the loss of information from using a diagonal matrix with the potential for overfitting that comes from using the full covariance matrix.

6.0.2 Runtimes

As we have claimed earlier, our motivation is to improve approximation without incurring large computational cost. We timed the training phase of each of the experiments. Figures 4 to 6 graph these runtimes with respect to sample dimension.

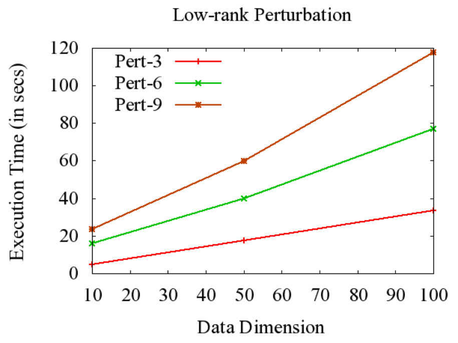


Figure 4 Relation between runtimes and sample dimension for Low Rank Perturbed

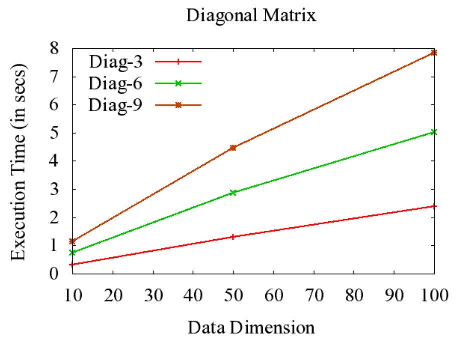


Figure 5 Relation between runtimes and sample dimension for Diagonal

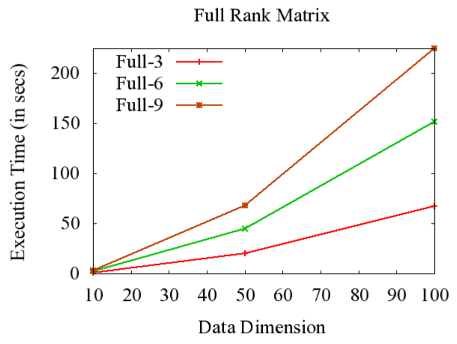


Figure 6 Relation between runtimes and sample dimension for Full Covariance Matrix

Since the scale of time is not the same, the runtimes for each approximation are plotted separately. Each graph shows how the training time increases with dimension for models with various numbers of mixtures (in this case, 3,6, and 9 mixtures). This makes it easier to see how the computational cost increases with respect to sample dimension. In these graphs, we can see that both the diagonal and perturbed approximations have a cost that increases linearly with respect to sample dimension, since they are bounded by $O(d)$. Also, we see the full covariance matrix cost increases quadratically. Table 5 shows the ratio of runtimes between using the full matrix to the perturbed matrix clearly growing with dimension.

6.1 Experimental Results on Real Data

We look at the effect of our low rank perturbed approximation on training GMMs on real world speech. The speech comes from a TIMIT corpus of American English accents. Speakers who were raised in various places in America were recorded while speaking 10 sentences. We look only at those speakers who were raised in the northern and southern regions.

The data is processed so that every 25ms of speech yields a 39-dimensional sample. We then train two GMMs, one for the northern samples and another for

	Full Cov.	Diagonal	Perturbed
Sample Acc.	61.6%	49.5%	54.3%
Word Acc.	76.5%	45.9%	60.0%

Table 6 Percentage accuracy of GMMs on speech samples

the southern samples. Test samples are then categorized based on which GMM gives a higher log probability. Whole words are similarly classified based on which GMM gives a higher average log probability.

6.1.1 Training Performance

Table 6 shows the resulting accuracy of using various approximations for the covariance matrix when classifying speech samples. As we saw in the synthetic data, the low rank perturbed approximation gives a better performance over the diagonal approximation. In particular, there is a significant increase with respect to the accuracy in classifying spoken words.

7 Conclusion

We have proposed a new method of approximating the covariance matrix for a Gaussian Mixture Model. Instead of simply using the diagonal, we use a low-rank perturbation of a diagonal matrix and approximate the inverse of the covariance matrix, rather than the matrix itself. The conjugate gradient method is used to optimize the parameters with respect to the log probability formula. We have developed a training algorithm that is bounded linearly with respect to sample dimension. This makes its computational cost comparable to that of the diagonal approximation, while at the same time able to model correlations in the data.

In our experimental results we compared our approximation method to the diagonal approximation and the full covariance matrix. We have shown that our matrix approximations not only outperform the diagonal approximation but avoid the overfitting that comes with the full rank covariance matrix. Further, we presented runtimes that demonstrates the comparable computation cost of the low-rank approximation to the diagonal approximation. The additional cost of the low-rank approximation over the diagonal approximation is acceptable in any situation where the variances are insufficient (as seen with the alphabet dataset).

It is of theoretical interest to develop analytic solutions to the LRPDD problem. This matrix decomposition may also be of independent interest. While we have a working version of the LRGMM, there are a few ways to improve upon it. Currently we use the conjugate gradient to improve the approximation. Finding appropriate stopping criteria can further improve the runtime of the LRGMM. Also, the LRGMM uses a rank 1 perturbation. It is possible to create an adaptive LRGMM that adds additional ranks of perturbations. Finally, as previously stated, the overlapping clusters are determined by a simple heuristic. There is promise in exploring alternative heuristics for assigning data points to clusters. The code for this model can be found bundled with our social networking code, which is available online by Purnell (2010).

ACKNOWLEDGEMENTS

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Magnus Astrand, Petter Mostad, and Mats Rudemo. Improved covariance matrix estimators for weighted analysis of microarray data. *J. Comp Bio. : a journal of computational molecular cell biology*, 14(10):1353–67, 2007. ISSN 1066-5277. doi: 10.1089/cmb.2007.0078.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

S. Chaudhuri, M. Drton, and T. S. Richardson. Estimation of a covariance matrix with zeros. *Biometrika*, 94(1):199–216, 2007. ISSN 0006-3444. doi: 10.1093/biomet/asm007.

Sanjoy Dasgupta and Leonard J. Schulman. A two-round variant of em for gaussian mixtures. In *Proc 16th Conference on Uncertainty in Artificial Intelligence*, pages 152–159, 2000.

Noureddine El Karoui. Spectrum estimation for large dimensional covariance matrices using random matrix theory. *The Annals of Statistics*, 36(6):2757–2790, 2008. ISSN 0090-5364. doi: 10.1214/07-AOS581.

Vikas K. Garg and M.N. Murty. Feature subspace svms fs-svms for high dimensional handwritten digit recognition. *International Journal of Data Mining, Modelling and Management*, 1(4):411–436, 2009.

Wouter J. Den Haan and Andrew Levin. A Practitioner's Guide To Robust Covariance Matrix Estimation. In *in Handbook of Statistics: Robust Inference (Vol 15*, pages 291–341. North-Holland, 1997. doi: 10.1.1.41.4595.

Magnus Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6), 1952.

Liping Jing, Junjie Li, Michael K. Ng, Yiu-ming Cheung, and Joshua Huang. Smart: a subspace clustering algorithm that automatically identifies the appropriate number of clusters. *International Journal of Data Mining, Modelling and Management*, 1(2): 149–177, 2009.

M. Pourahmadi. Cholesky Decompositions and Estimation of A Covariance Matrix: Orthogonality of Variance Correlation Parameters. *Biometrika*, 94(4):1006–1013, 2007. ISSN 0006-3444. doi: 10.1093/biomet/asm073.

Jonathan Purnell. LRGMM code. *http://www.cs.rpi.edu/˜purnej/code.php*, 2010.

Chafika Ramdane, Souham Meshoul, Mohamed Batouche, and Mohamed-Khireddine Kholladi. A quantum evolutionary algorithm for data clustering. *International Journal of Data Mining, Modelling and Management*, 2(4):369–387, 2010.

Renxia Wan, Jingchao Chen, Lixin Wang, and Xiaoke Su. Grid-based clustering over an evolving data stream. *International Journal of Data Mining, Modelling and Management*, 1(4):393–410, 2009.