Approximating the Covariance Matrix with Low-rank Perturbations

Malik Magdon-Ismail and Jonathan T. Purnell

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
{magdon,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. We present an algorithm that improves upon the diagonal matrix by allowing a low rank perturbation. The efficiency is comparable to the diagonal approximation, yet one can capture correlations among the dimensions. We show that this method outperforms the diagonal when training GMMs on both synthetic and real-world data.

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

1 Introduction

Sample covariance matrices provide important information about the probability distribution of the samples. This information can be used in pre-processing, such as PCA and whitening, where the samples can be reduced in dimensionality or decorrleated. Our focus is on fitting probabilistic models, such as the Gaussian Mixture Model. In particular, we will be looking at high dimensional models.

A major drawback of the covariance matrix is its computational cost, which grows quadratically with respect to the sample dimension. Thus, as the dimension of our data grows, the curse of dimensionality comes into effect. Although there are methods for feature selection or dimension reduction, it is not always desirable or sufficient. Leaving out samples does not help much since the computational cost grows linearly with respect to our sample size. The additional drawback of the full covariance matrix in high dimension is that the additional $O(d^2)$ parameters can lead to heavy overfitting.

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 will explore further in Section 2, there are several proposed approximations which mostly utilize decompositions of the covariance matrix.

While the diagonal can be computed quickly, it loses all the correlation information. We propose to use the diagonal plus a low-rank perturbation. The motivation is to keep the computational cost linearly bounded to the dimension, while obtaining correlation information; this yields a more accurate approximation by only using twice the number of parameters, d parameters for the diagonal and d for the perturbation. These extra parameters provide an improvement over the high restrictions of the diagonal. Note that the low rank perturbation does not yield a low rank covariance matrix. The perturbation is what will be low rank, but will add back most of the correlation.

Our method is tested against the diagonal approximation and the full covariance matrix for Gaussian Mixture Models. Our experiments involve various types of synthetic data as well as real world data. We examine the in-sample and out-sample average log probabilities of the models. The runtime is compared between the models to show that it is bounded linearly with respect to the sample dimension. The practicality of our method is shown by its application to real world data.

2 Covariance Matrix Approximation

We seek to address two problems. First is the computational cost of training a Gaussian Mixture Model (GMM) when using the full covariance matrix. Second is the potential to overfit due to $O(d^2)$ parameters in the full covariance matrix.

Problem Definition The covariance is the defining characteristic of the GMM over other clustering algorithms. 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. The maximization step updates the covariance matrix using these probabilities. The running time of a single step of this EM algorithm is $O(NKd^2)$, where N is the number of samples and K is the number of mixtures in the model. The appearance of d^2 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 of the GMM.

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

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

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \Sigma = \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)(x_i-\mu)^T \ . \qquad (2)$$

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.

Previous Work There has been some work in covariance matrix approximations. These involve a variety of approaches, mainly decomposition [3][7], statistical estimation [4], and using assumptions on the matrix [2][1].

In [3], El Karoui looks into estimating the spectrum of large dimensional covariance matrices. The vector of eigenvalues is defined as an observation of a probability distribution. Using random matrix theory and convex optimization, this probability distribution is estimated. The resulting estimates can be used in algorithms such as PCA. In [7], decompositions are used to overcome the difficulty of estimating positive definite matrices. The Cholesky decomposition is seen as a better decomposition than variance-correlation or spectral decomposition. In [4], authors use kernel-based and parametric spectral estimation procedures to produce heteroskedasticity and autocorrelation consistent matrices. These above methods have the advantage of being applicable to most types of covariance matrices, but suffer by having quadratic, or worse, time complexities. The following approaches exploit various properties to achieve better runtimes.

In [2], an iterative method of estimation is proposed using an equivalent covariance graph and constraining the covariance matrix to be sparse. In [1], the authors estimate covariance matrices of particular distributions by two methods. First, by the EM algorithm. Secondly, by using a priori information on the distribution of the samples. With limited data, [6] finds that a mixture of the sample and common covariance matrices, along with their diagonals, can be a better approximation of the covariance matrices. In general, all these methods are $O(NKd^2)$ or cannot be efficiently inverted.

3 Approximating the Full Covariance

The diagonal approximation is the simplest. Let $\Delta x_n = x_n - \mu$ and $\Sigma = \frac{1}{N}\sum_i \Delta x_i \Delta x_i^T$; for $\hat{\Sigma} = D$, a diagonal matrix,

$$\log \mathbf{L} = -\frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^T D^{-1}(x_n-\mu) + \frac{N}{2}\log|D^{-1}| - \frac{Nd}{2}\log 2\pi \ , \tag{3}$$

which is maximized when we minimize

$$\varepsilon = \frac{1}{N}\sum_{n=1}^{N}\Delta x_n^T D^{-1}\Delta x_n - \log|D^{-1}| = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{d}\frac{\Delta x_n(i)^2}{D_{ii}} - \sum_{i=1}^{d}\log\frac{1}{D_{ii}} \ . \tag{4}$$

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

$$\frac{\partial \varepsilon}{\partial D} = \frac{1}{N}\sum_{n=1}^{N}\Delta x_n^2 - D \Rightarrow D = \frac{1}{N}\sum_{n=1}^{N}\Delta x_n^2 \ . \tag{5}$$

We can see that, with the diagonal constraint, the maximizer of $\log \mathbf{L}$ is the diagonal of Σ. This can be computed in $O(Nd)$, but loses all off-diagonal correlations.

3.1 Low Rank Perturbation to Diagonal

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

$$\Sigma^{-1} \approx D^2 + aa^T \ . \tag{6}$$

where D is our diagonal matrix and a is a d-dimensional vector that defines the low-rank perturbation. D^2 is used to ensure the approximation is positive semi-definite without having to add constraints The log-likelihood of a GMM, using this approximation is

$$\begin{aligned}\log \mathbf{L} = &-\frac{1}{2}\sum_{i=1}^{N}(x_i-\mu)^T(D^2+aa^T)(x_i-\mu) \\ &+\frac{N}{2}\log|(D^2+aa^T)| - \frac{Nd}{2}\log 2\pi \ .\end{aligned} \tag{7}$$

where N is the number of samples and d is the sample dimension. Maximization of this equation is equivalent to minimizing our optimization equation:

$$\varepsilon = \min \frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^T(D^2+aa^T)(x_i-\mu) - \log|(D^2+aa^T)| \ . \tag{8}$$

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 approximation's d parameters. Therefore, we expect the computational cost to be higher than the diagonal approximation, but still below that of the full-rank approximation which has $d(d+1)/2$ parameters. Although this new approximation may seem nearly as restrictive as the diagonal, the addition of off-diagonal information allows for a significantly closer approximation. 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. In this paper we explore a rank-1 perturbation.

In finding the optimal values for D and a, it may seem that using the diagonal of Σ for D is sufficient. However, this requires fitting aa^T to Σ with zeros along the diagonal. This pulls a away from its optimal value and towards a saddle-point at $a = 0^d$. In finding the optimal values for our parameters, D and 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:

$$\begin{aligned}\frac{\partial\varepsilon}{\partial D_\alpha} &= \frac{2}{N}\sum_{n=1}^{N}x_{n\alpha}^T D_\alpha x_{n\alpha} - \text{trace}\left((D^2+aa^T)^{-1}\frac{\partial(D^2+aa^T)}{\partial D_\alpha}\right) \\ &= \sum_{n=1}^{N}x_{n\alpha}^T D_\alpha x_{n\alpha} - \text{trace}\left(\left((D^{-2} - \frac{D^{-2}aa^TD^{-2}}{1+a^TD^{-2}a}\right)(2\boldsymbol{\delta}_\alpha)\right) \\ &= \sum_{n=1}^{N}x_{n\alpha}^T D x_{n\alpha} - 2D_\alpha^{-1} + \frac{\text{trace}(2\boldsymbol{\delta}_\alpha D^{-2}aa^TD^{-2})}{1+a^TD^{-2}a} \ .\end{aligned} \tag{9}$$

Algorithm 1 Conjugate Gradient

```
1: **Input:** data $x$, diagonal matrix $D_0$, perturbation $a_0$, threshold $T_g$
2: $g_0 \leftarrow [\frac{\partial\varepsilon}{\partial a_0}, \frac{\partial\varepsilon}{\partial D_0}]$ ; $v_0 \leftarrow -g_0$.
3: **while** $|g_i| > T_g$ **do**
4:     Perform a line search along $v_i$ to find optimal step size, $s$.
5:     $a_{i+1} \leftarrow a_i + s \cdot v_i(1 \ldots d)$.
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: **end while**
```

where the derivative of the $\log |(D^2 + aa^T)|$ term is derived using Jacobi's formula $d \det(A) = \mathrm{tr}(\mathrm{adj}(A) dA)$; $\boldsymbol{\delta}_\alpha$ is the vector D with all elements set to zero except D_α; and $\mathrm{trace}()$ is the trace function. This formula can be written in vector form as:

$$\frac{\partial\varepsilon}{\partial D} = diag(\Sigma)D - 2D^{-1} + \frac{2d(D^{-2}a)^2}{1 + a^T D^{-2} a} \ . \tag{10}$$

The gradient for a can be derived in a similar manner

$$\frac{\partial\varepsilon}{\partial a_\alpha} = \frac{2}{N}\sum_{n=1}^{N}\left(\sum_{i=1}^{d} x_{ni}a_i\right)^2 x_{n\alpha} - \mathrm{trace}\left((D^2 + aa^T)^{-1}\frac{\partial(D^2 + aa^T)}{\partial a_\alpha}\right) \tag{11}$$

$$= \frac{2}{N}\sum_{n=1}^{N}\left(\sum_{i=1}^{d} x_{ni}a_i\right)^2 x_{n\alpha} - \frac{2a_\alpha}{D_\alpha} + \frac{2a_\alpha}{D_\alpha}\sum_{i=1}^{d}\frac{a_i^2}{D_i} \ . \tag{12}$$

which can be put into vector form as

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

Instead of an analytical solution, we use the conjugate gradient descent to reach an optimum[5] (See Algorithm 1). We initialize 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 a and the gradient for D. The search direction is referred to as v. The threshold, T_g, is set to 0.001, our stopping criterion.

Although in theory we would use d steps of the conjugate gradient method, practice shows that only a few steps are needed. In the next section, we explore how our approximation compares to the diagonal approximation and the full covariance matrix.

4 Experiments

Synthetic Data We synthesized GMM data 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 (full covariance matrix, diagonal, or low rank perturbed). See Table 4 for the results, which are averaged over 100 runs.

Training Performance We observe performance from two perspectives, the in-sample (samples used for training) and out-sample (samples used for testing) average log probabilities. Experiments were performed over various sample dimensions, dataset sizes, and number of model mixtures.

	In-Samp N=10k			In-Samp N=1k			Out-Samp N=1k		
Dims.	Diag.	Pert.	Full	Diag.	Pert.	Full	Diag.	Pert.	Full
10	-109	-101	-87	-10.9	-9.96	-8.53	-11.0	-10.2	-8.95
50	-1032	-1005	-734	-94.7	-93.2	-85.8	-95.6	-94.5	-95.6
100	-2250	-2219	-1667	-224.2	-225.6	-197.3	**-225.9**	**-232.2**	**-249.1**

Table 1. Probabilities for various sample sizes and dimensions

We see that the perturbed matrix probabilities are consistently closer to the full matrix probabilities than those for the diagonal matrix. Further, since this also holds in the out-sample results, we see that the perturbed matrix is not obtaining higher probabilities by overfitting the data. Note, however, that the full covariance matrix has the potential to severly overfit when d is large (see Table 4 for d = 100).

Runtimes Our main motivation is to improve approximation without incurring large computational cost. Run time for the training phase is shown in Figure 1.

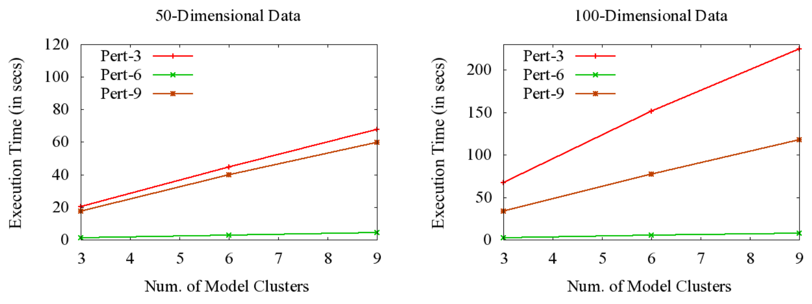

Fig. 1. Relation between runtimes and number of model clusters

In Figure 1, we compare the runtimes versus the number of clusters in our model. Here we can see that the approximation by perturbation is faster than the full rank matrix. This improvement becomes more significant with higher dimensional data and more complex models (i.e. more clusters). We see that the full rank performs almost as fast as our algorithm at low dimensions and low number of clusters. This is likely due to the overhead of the perturbation approximation, which could be overcome by further optimization. Also, it should be noted that in this case of low dimension and low model

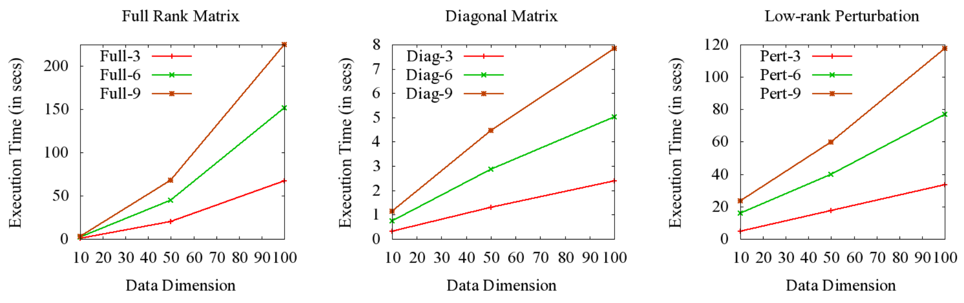

Fig. 2. Relation between runtimes and dimension.

complexity, the total computational time is also very low, so any approximation would most likely fail to improve performance enough to compensate for the lack of accuracy; our algorithm is most useful when d is large.

The runtimes in Figure 2, show 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; also, the full covariance matrix cost is quadratic.

4.1 Real Data

We look at GMMs for speech data. The speech comes from a TIMIT corpus of American English accents. Speakers who had grown up from various place in America were recorded while speaking 10 sentences. We look particularly 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 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.

The out-sample results are show in the table to the right. 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.

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

Fig. 3. Accuracy of GMMs on speech samples

5 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. The conjugate gradient method is used to optimize the parameters with respect to the log probability of a GMM. By approximating the inverse of the covariance matrix, we developed an algorithm that is bounded linearly with respect to sample dimension. This makes its computational cost comparable to the that of the diagonal approximation.

We compared our approximation method to the diagonal approximation and the full covariance matrix. The in-sample accuracies show that while the perturbed matrix is not an exact approximation, it is consistently better than the diagonal. Further, the out-sample accuracy shows that this improved flexibility does not result in overfitting, since the perturbed approximation also out-performs the diagonal approximation in this respect as well. Our method has a comparable computation cost to the diagonal approximation (linear in d).

Although more complex in theory, the implementation of our method is efficient, since the conjugate gradient method has very efficient implementations. Also, this method is applicable to any sample-based covariance matrix, and not specifically in a GMMs.

In the future, we may look into lowering the absolute computational cost of the perturbed approximation by finding the optimal values and thresholds for the conjugate gradient method. Also, we plan to look into using various ranks of perturbation. Intuitively, it seems that all perturbations from rank 1 to full rank can be similarly derived and provide a greater flexibility in finding the balance between computational cost and accuracy. This would allow the approximation to be more adaptable.

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

1. Astrand, M., Mostad, P., Rudemo, M.: 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)
2. Chaudhuri, S., Drton, M., Richardson, T.S.: Estimation of a covariance matrix with zeros. Biometrika 94(1), 199–216 (2007)
3. El Karoui, N.: Spectrum estimation for large dimensional covariance matrices using random matrix theory. The Annals of Statistics 36(6), 2757–2790 (2008)
4. Haan, W.J.D., Levin, A.: A Practitioner's Guide To Robust Covariance Matrix Estimation. In: in Handbook of Statistics: Robust Inference (Vol 15. pp. 291–341. North-Holland (1997)
5. Hestenes, M., Stiefel, E.: Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards 49(6) (1952)
6. Hoffbeck, J., Landgrebe, D.: Covariance matrix estimation and classification with limited training data. IEEE Tran Pattern Anal and Mach Intel 18(7), 763–767 (1996)
7. Pourahmadi, M.: Cholesky Decompositions and Estimation of A Covariance Matrix: Orthogonality of Variance Correlation Parameters. Biometrika 94(4), 1006–1013 (2007)