Adaptive Riemannian stochastic gradient descent and reparameterization for Gaussian mixture model fitting

Chunlin Ji✉ CHUNLIN.JI@KUANG-CHI.ORG
Kuang-Chi Institute of Advanced Technology, Shenzhen, China

Yuhao Fu YUHAO.FU@KUANG-CHI.COM
Kuang-Chi Institute of Advanced Technology, Shenzhen, China
Origin Artificial Intelligence Technology Co., Shenzhen, China

Ping He SABER@SSR.FUND
HeGuangLiangZi Tech., Shenzhen, China

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Recent advances in manifold optimization for the Gaussian mixture model (GMM) have gained increasing interest. In this work, instead of directly addressing the manifold optimization on covariance matrices of GMM, we consider the GMM fitting as an optimization of the density function over a statistical manifold and seek the natural gradient to speed up the optimization process. We present an upper bound for the Kullback–Leibler (KL) divergence between two GMMs and obtain simple closed-form expressions for the natural gradients. With the natural gradients, we then apply the Riemannian stochastic gradient descent (RSGD) algorithm to optimize covariance matrices on a symmetric and positive definite (SPD) matrix manifold. We further propose a Riemannian Adam (RAdam) algorithm that extends the momentum method and adaptive learning in the Euclidean space to the SPD manifold space. Extensive simulations show that the proposed algorithms scale well to high-dimensional large-scale datasets and outperform expectation maximization (EM) algorithms in fitted log-likelihood.

Keywords: Gaussian mixture model; Reparameterization; Symmetric positive definite matrix manifold; Riemannian stochastic gradient descent; Riemannian Adam algorithm

1. Introduction

Gaussian mixture model, also called a multivariate normal mixture, is a powerful statistical tool that could be used to approximate any density defined on $\mathbb{R}^d$ with a large enough number of mixture components. GMM is widely used as a flexible model with various successful applications in speech recognition (Povey et al., 2010), image representation (Beecks et al., 2011) and time series classification (Campbell et al., 2006) among many others.

GMM fitting is commonly solved by the expectation maximization (EM) algorithm (Dempster et al., 1997). However, EM suffers from the local optimal of the likelihood function (Jin et al., 2016) and has its speed limits for highly overlapping clusters (Xu and Jordan, 1996). Recently, alternative approaches have tried to address the GMM fitting from a nonlinear optimization perspective and solving it by the manifold optimization Absil

et al. (2009). Although several manifold optimization approaches, such as Riemannian LBFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) (Hosseini and Sra, 2015; Godaz et al., 2021) and Riemannian Newton trust region (Sembach et al., 2022) have been proposed and show promising results, stochastic gradient descent (SGD) type algorithms have not been thoroughly studied.

In this work, we explore the Riemannian SGD-type algorithm for GMM fitting. We follow the reparameterization strategy for Gaussian mixtures introduced in Hosseini and Sra (2015), which alleviates the dependence when iteratively estimating the mean and covariance of a GMM and allows us to focus only on estimating the covariance matrices. Instead of directly addressing the manifold optimization on covariance matrices, we consider the GMM fitting as an optimization of the probability density function defined over a statistical manifold and seek the natural gradient, a second-order optimization method (Lin et al., 2019; Martens, 2020), to speed up the optimization process. We present an upper bound for the KL divergence between two GMMs and obtain simple closed-form expressions for the natural gradients. Given the natural gradients, we apply the Riemannian stochastic gradient descent (RSGD) algorithm to optimize the covariance matrices while preserving their positive definiteness constraint. Furthermore, we propose a Riemannian Adam (RAdam) algorithm, which employs past gradients to compute a local distance measure on the SPD matrix manifold and subsequently rescales the learning rate to gain faster convergence. Experimental studies show that both the proposed RSGD and RAdam methods scale well to large-scale high-dimensional datasets. The RAdam algorithm generally obtains a better log-likelihood than the RSGD and is more robust to the learning step size. Compared to EM, we find that the RSGD and RAdam gain significantly better log-likelihood for most datasets, including several challenging scenarios for GMM fitting.

2. Related Works

Using gradient-based optimization to learn the parameter of a statistical model has become a popular approach for model inference. Unlike a simple vector in Euclidean space, the parameter of a statistical model may have an intrinsic structure. Observing the specialty of the Riemannian manifold structure on a statistical model, we can take advantage of the natural gradient descent method (Amari, 1997), a second-order optimization method, to improve convergence. It has recently gained increasing interest from various areas such as variational inference (Lin et al., 2019) and reinforcement learning (Schulman et al., 2015). In case the statistical model is an exponential family distribution, the derivation of natural gradient is effective, as the required Fisher Information Matrix (FIM) has a simple explicit form. However, for a mixture distribution, such as a GMM, the computation of FIM becomes challenging. Recently, Lin et al. (2019) used a minimal conditional exponential family representation for the mixtures and obtained a natural gradient update. In this work, we propose a simple alternative to obtain the natural gradient by using an upper bound of the KL divergence instead of directly computing the FIM from the KL divergence.

When optimizing the parameters of a statistical model, we always need to consider the constraint on these parameters, for example, guaranteeing that the covariance matrix is SPD. To this end, manifold optimization (Absil et al., 2009) is an elegant way to fit this goal by processing the model parameter in a manifold space (Hosseini and Sra, 2015; Kasai et al.,

2019; Tran et al., 2021). Hosseini and Sra (2015) introduce a reparameterization strategy for the Gaussian distribution, resulting in a simple model where we only need to optimize the mixture weights and the augmented covariance matrix. They propose a Riemannian LBFGS (RLBFGS) to optimize the covariance matrix on the SPD manifold. However, the RLBFGS is relatively complex for implementation and requires heavy computation to calculate Hessian matrices. In a following-up work, Godaz et al. (2021) proposes a vector transport free Riemannian LBFGS, for matrix manifold optimization, which simplifies the computation but the performance is similar to that of the previous manifold LBFGS. In addition, Tran et al. (2021) use manifold optimization for the covariance of the Gaussian distribution in a variational inference setting; however, they do not mention their method for GMM fitting. In a closely related work, Lin et al. (2020) uses a retraction on the SPD matrix manifold to update the covariance matrices to handle the positive constraint. In this work, we leverage advantages of the Riemannian SGD algorithm (Bonnabel, 2013), which is a generalization of the (Euclidean) SGD algorithm to Riemannian manifolds. Moreover, we develop an Adam-style algorithm, which significantly improves the convergence rate.

3. Proposed Methods

3.1. Problem setup

Let $x \in \mathbb{R}^d$ denote the variable whose distribution, denoted as $\pi(x)$, is the target we want to infer, and generally a set of i.i.d. samples $\{x_1, ..., x_n\}$ are given. Our goal in this paper is to approximate the target distribution $\pi(x)$ with a Gaussian Mixture Model (GMM), whose probability density is $q(x) = \sum_{k=1}^{K} w_k \mathcal{N}(\cdot; \mu_k, \Sigma_k)$, where $\sum_{k=1}^{K} w_i = 1$ and $\mathcal{N}(x; \mu, \Sigma)$ is a (multivariate) Gaussian with mean $\mu \in \mathbb{R}^d$ and covariance $\Sigma \succ 0$, where $\cdot \succ 0$ denote positive definite. Given the samples $\{x_1, ..., x_n\}$, we want to estimate the parameter $\{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$. Generally, we use the the marginal log-likelihood as the optimization criterion for model fitting,

$$\mathcal{L}(\{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}) = \frac{1}{n} \sum_{i=1}^{n} \log \sum_{k=1}^{K} w_k \mathcal{N}(x_i; \mu_k, \Sigma_K). \tag{1}$$

3.2. Reparameterization

EM and many of its variants directly optimize the parameter μ and Σ respectively; however, the estimation of Σ depends on the estimation of μ, particularly when using a batch data for training μ, the variance of estimated μ becomes larger and therefore causes negative effects on the estimation of Σ. Following the previous work (Hosseini and Sra, 2015), we reparameterize the GMM, which leads to a way to optimize μ and Σ simultaneously.

Taking a single Gaussian as an example, we augment the sample vectors x_i by an extra dimension and consider $y_i = [x_i^T, 1]^T$; correspondingly, we can transform the marginal log-likelihood of a single Gaussian estimation, $\mathcal{L}(\mu, \Sigma) := \frac{1}{n} \sum_{i=1}^{n} \log \mathcal{N}(x_i; \mu, \Sigma)$ into $\mathcal{L}(S) = \frac{1}{n} \sum_{i=1}^{n} \log q_{\mathcal{N}}(y_i; S)$, where $q_{\mathcal{N}}(y_i; S) := 2\pi \exp(\frac{1}{2}) \mathcal{N}(y_i; S)$, S is the augmented covariance matrix. Hosseini and Sra (2015) has been proved that if μ^* , Σ^* maximize $\mathcal{L}(\mu, \Sigma)$, and if S^* maximizes $\mathcal{L}(S)$, then $\mathcal{L}(S^*) = \mathcal{L}(\mu^*, \Sigma^*)$ for $S^* = \begin{pmatrix} \Sigma^* + \mu^* \mu^{*T} & \mu^* \\ \mu^* & 1 \end{pmatrix}$.

Similarly, we can reparameterize the GMM model, and transform Eq.(1) into

$$\mathcal{L}(\{w_j, S_j\}_{j=1}^K) = \frac{1}{n}\sum_{i=1}^{n} \log\left(\sum_{j=1}^{K} w_j q_{\mathcal{N}}(y_i; S_j)\right). \tag{2}$$

The local maximum of the reparameterized GMM log-likelihood (Eq.(2)) is also a local maximum of the original log-likelihood (Eq.(1)).

Given the function $\mathcal{L}(\cdot)$, we expect to optimize the model parameter $\phi := \{w_j, S_j\}_{j=1}^K$ with the SGD type algorithm. Supposing the batch data $\{y_i\}_{i=1}^{N_s}$ is given, we can derive the gradient of $\mathcal{L}$ with respect to w_k and S_k respectively as follows (detailed in the Appendix),

$$\nabla_{w_k}\mathcal{L} = \frac{1}{N_s}\sum_{i=1}^{N_s} \frac{q_{\mathcal{N}}(y_i; S_k)}{\boldsymbol{q}_\phi(y_i)} - 1, \tag{3}$$

$$\nabla_{S_k}\mathcal{L} = \frac{1}{N_s}\sum_{i=1}^{N_s} \frac{w_k q_{\mathcal{N}}(y_i; S_k)}{\boldsymbol{q}_\phi(y_i)} \frac{1}{2}\left(S^{-1}y_i y_i^\top S^{-1} - S^{-1}\right), \tag{4}$$

where we denote $\boldsymbol{q}_\phi(\cdot) = \sum_{j=1}^K w_j q_{\mathcal{N}}(\cdot; S_j)$ for simplicity.

3.3. Natural Gradient Descent

When we optimize the parameter ϕ of a distribution q_ϕ, we expect small distances between the statistical manifold of q_ϕ and $q_{\phi'}$ rather than small naive Euclidean distances between ϕ and ϕ'. The purpose of natural gradient descent is to take advantage of the information geometry of q_ϕ to accelerate convergence. To optimize over a manifold, we need to modify our Euclidean gradients using an appropriate local scaling, also known as a Riemannian metric (Amari, 1997). A common Riemannian metric for statistical manifolds is the Fisher Information Matrix (FIM), which is defined as $F_\phi = E_{q_\phi(\cdot)}[\nabla_\phi \log q_\phi(\cdot)(\nabla_\phi \log q_\phi(\cdot))^T]$. Correspondingly, the natural gradient can be obtained by $\nabla_\phi^{\text{nat}}\mathcal{L} = F_\phi^{-1}\nabla_\phi\mathcal{L}$ (detailed in the Appendix). The natural gradient $\nabla_\phi^{\text{nat}}\mathcal{L}$ corresponds to the direction of the steepest ascent along the statistical manifold $q_\phi(\cdot)$. The preconditioning of the gradients F_ϕ^{-1} leads to a proper local scaling of the gradient in each dimension and takes into account the dependencies between variables. This always leads to faster convergence in optimizing q_ϕ.

When q_ϕ is a single Gaussian, the Fisher information matrix for the multivariate Gaussian distribution $N(\mu, \Sigma)$ is $F_\phi = \begin{pmatrix} \Sigma^{-1} & 0 \\ 0 & I_F(\Sigma) \end{pmatrix}$ where $I_F(\Sigma) \approx \Sigma^{-1} \otimes \Sigma^{-1}$, with $\otimes$ denoting the Kronecker product. Therefore, $F_\phi^{-1} \approx \begin{pmatrix} \Sigma & 0 \\ 0 & \Sigma \otimes \Sigma \end{pmatrix}$, which gives a convenient form for obtaining an approximate natural gradient. For the reparameterized Gaussian $q_{\mathcal{N}}(y_i; S) := 2\pi\exp(\frac{1}{2})\mathcal{N}(y_i; S)$, the natural gradient of $q_{\mathcal{N}}(y_i; S)$ w.r.t S becomes,

$$\nabla_S^{\text{nat}}\mathcal{L} = \text{vec}^{-1}\left((S \otimes S)\nabla_{\text{vec}(S)}\mathcal{L}\right) = 2S\left(\nabla_S\mathcal{L}\right)S \tag{5}$$

$$= 2S\left(\frac{1}{N_s}\sum_{i=1}^{N_s}\frac{1}{2}\left(S^{-1}y_i y_i^\top S^{-1} - S^{-1}\right)\right)S = \frac{1}{N_s}\sum_{i=1}^{N_s}\left(y_i y_i^\top - S\right), \tag{6}$$

where $\text{vec}(X)$ denotes the vector obtained by stacking the columns of X below one another; $\text{vec}^{-1}(X)$ denotes the inverse processes of $\text{vec}(X)$ that $\text{vec}^{-1}(\text{vec}(X)) = X$.

3.4. Gaussian Mixture updates

In this work, q_ϕ is a mixture of multivariate normal distribution. Unfortunately, it is not easy to find a simple closed-form expression for FIM in this case. Here, instead of using the KL divergence directly, we seek some bounds for approximation. We utilize an upper bound of the KL divergence between the two Gaussian mixture densities, $\boldsymbol{q}_\phi = \sum_{k=1}^K w_k q_\mathcal{N}(\cdot; S_k)$ and $\boldsymbol{q}_{\phi'} = \sum_{k=1}^K w'_k q_\mathcal{N}(\cdot; S'_k)$ which is given as follows (Hershey and Olsen, 2007),

$$KL[\boldsymbol{q}_\phi || \boldsymbol{q}_{\phi'}] \leq \sum_{k=1}^K w_k KL[q_\mathcal{N}(\cdot; S_k) || q_\mathcal{N}(\cdot; S'_k)] + KL[\boldsymbol{w} || \boldsymbol{w}'], \tag{7}$$

where $\boldsymbol{w} = [w_1, ..., w_K]^T$ and $KL[\boldsymbol{w}||\boldsymbol{w}'] = \sum_{k=1}^K w_k \log(w_k/w'_k)$. Here, we use the upper bound, $KL[\boldsymbol{w}||\boldsymbol{w}'] + \sum_{k=1}^K w_k KL[q_\mathcal{N}(\cdot; S_k)||q_\mathcal{N}(\cdot; S'_k)]$, as the regularization term instead of $KL[\boldsymbol{q}_\phi||\boldsymbol{q}_{\phi'}]$, that we keep this upper bound as a constant in deriving the natural gradients.

As shown in the last section, the KL divergence between the two single multivariate normal distributions has led to a closed-form solution for natural gradients. The upper bound with a mixture of KL divergence between individual $q_\mathcal{N}(\cdot; S_k)$ and $q_\mathcal{N}(\cdot; S'_k)$ (for $k = 1, ..., K$) can also lead to a closed-form solution for the natural gradient (refer to the Appendix for detailed derivation). The resulting natural gradients for all the parameters $\{w_k, S_k\}_{k=1}^K$ in the reparameterized GMM are

$$\nabla^{\text{nat}}_{w_k} \mathcal{L} = w_k \nabla_{w_k} \mathcal{L} = \frac{1}{N_s} \sum_{i=1}^{N_s} \frac{w_k q_\mathcal{N}(y_i; S_k)}{\boldsymbol{q}_\phi(y_i)} - w_k, \tag{8}$$

$$\nabla^{\text{nat}}_{S_k} \mathcal{L} = \frac{1}{w_k} 2 S_k \left(\nabla_{S_k} \mathcal{L}\right) S_k = \frac{1}{w_k N_s} \sum_{i=1}^{N_s} \frac{w_k q_\mathcal{N}(y_i; S_k)}{\boldsymbol{q}_\phi(y_i)} \left(y_i y_i^\top - S_k\right). \tag{9}$$

3.5. Riemannian stochastic gradient descent

To update a covariance matrix S with gradients, we need to ensure that the resulting matrix is SPD. Riemannian SGD-style algorithms (Bonnabel, 2013) have been widely applied for optimization on various manifolds, including the SPD matrix manifold. To begin with such an algorithm, we briefly introduce some concepts of Riemannian geometry and optimization. The key Riemannian operation for the SPD matrix manifold is also provided in the Appendix. We refer interested readers to Absil et al. (2009) for more details.

Riemannian manifold. A Riemannian manifold $(\mathcal{M}, \rho)$ is a smooth manifold $\mathcal{M}$ equipped with a Riemannian metric ρ defined as the inner product on the tangent space $\Gamma_S\mathcal{M}$ for each point S, $\rho_S(\cdot, \cdot) : \Gamma_S\mathcal{M} \times \Gamma_S\mathcal{M} \to \mathbb{R}$. The tangent space $\Gamma_S\mathcal{M}$ is the first order approximation of $\mathcal{M}$ around a point $S \in \mathcal{M}$. The tangent space $\Gamma_S\mathcal{M}$ contains all tangent vectors to $\mathcal{M}$ at S. In this work, we study the manifold $\mathcal{M}$ of SPD matrices $\mathcal{M} = \{S \in \text{Mat}(d+1, d+1) : S = S^T, S \succ 0\}$, with $\rho_S(\xi, \eta) = tr(S^{-1}\xi S^{-1}\eta)$.

Exponential map and Retraction. Exponential maps are mappings that, given a point S on a manifold $\mathcal{M}$ and a tangent vector $\xi_S \in \Gamma_S\mathcal{M}$ at S, generalize the concept $S + \xi_S$ in Euclidean spaces. $Exp_S(\xi_S)$ is a point on the manifold that can be reached by leaving from and moving in the direction ξ_S while remaining on the manifold. As computing

an exponential map is usually expensive, a retraction map is often used as an efficient alternative. The SPD matrix manifold has a simple closed-form formula for retraction,

$$R_S(\alpha\xi_S) = S + \alpha\xi_S + \frac{\alpha^2}{2}\xi_S S^{-1}\xi_S, \xi_S \in \Gamma_S\mathcal{M}. \tag{10}$$

Parallel transport and vector transport. Parallel transport is a method to translate tangent vectors from one tangent space to another, while still preserving the length and angle of the original tangent vectors. Vector transport is a computationally efficient alternative to parallel transport. Let $\Gamma_{S_t\to S_{t+1}}(\xi_{S_t})$ denote the vector transport of the tangent vector $\xi_{S_t} \in \Gamma_{S_t}\mathcal{M}$ to the tangent space $\Gamma_{S_t+1}\mathcal{M}$. For the SPD matrix manifold, the vector transportation also takes a simple closed-form formula,

$$\Gamma_{S_t\to S_{t+1}}(\xi_{S_t}) = E\xi_{S_t}E^T, E = (S_{t+1}S_t^{-1})^{1/2}. \tag{11}$$

3.6. The proposed algorithms

The RSGD extends the traditional SGD gradient update in the Euclidean space to the Riemannian space by (Bonnabel, 2013)

$$S_{t+1} = R_{S_t}(-\alpha_t\mathcal{P}_S(\nabla_{S_t}\mathcal{L})), \tag{12}$$

where $\alpha_t > 0$ is a (decaying) step size. The orthogonal projection $\mathcal{P}_S(\nabla_{S_t}\mathcal{L})$ transforms the gradient $\nabla_{S_t}\mathcal{L}$, the Euclidean gradient with respect to S_t calculated at time t, into the tangent space $\Gamma_S\mathcal{M}$. For the SPD matrix manifold, the orthogonal projection $\mathcal{P}_S(\nabla_{S_t}\mathcal{L}) = \frac{1}{2}S\left(\nabla_{S_t}\mathcal{L} + [\nabla_{S_t}\mathcal{L}]^T\right)S$ is coincident with the natural gradient $\nabla_S^{\text{nat}}\mathcal{L}$ in Eq.(9). In addition, the retraction $R_S(\cdot)$ is given in Eq.(10). The detailed RSGD algorithm for GMM fitting is provided in the Appendix.

As one of the most successful variants of SGD, the Adam algorithm (Kingma and Ba, 2014) update rule for a scaler variable x is given by $x_{t+1} \leftarrow x_t - \alpha m_t/\sqrt{v_t}$, where g_t is the gradient, $m_t = \beta_1 m_{t-1} + (1-\beta_1)\, g_t$ is a momentum term and $v_t = \beta_2 v_{t-1} + (1-\beta_2)\,(g_t)^2$ is an adaptivity term. Similarly to the Adam in Euclidean space, we propose the Riemannian Adam (RAdam) for the optimization of the SPD matrix manifold. Extending the momentum term in Euclidean space to a manifold space requires the vector transportation defined in Eq.(11). The term $(g_t)^2$ in the Euclidean space is extended to the square of the Frobenius norm of a SPD matrix $||S||_F^2 = tr(S^TS)$. In addition, the natural gradients in Eqs.(8-9) (the gradient on the tangent space $\Gamma_S\mathcal{M}$) and retraction update Eq.(10) are also required. Specifically, the RAdam algorithm is shown in Algorithm 1.

The learning rate (or step size) α_t is another important hyperparameter in manifold optimization. Unlike using the Armijo or Wolfe condition to find the best step size in traditional manifold optimization (Hosseini and Sra, 2015), the SGD/Adam method allows us to choose a predefined learning rate, such as a small constant. In our problem setting, we find that a sequence of decreasing learning rates is easier to obtain a faster convergence than a constant one. Here, we provide some reduction strategies, such as $g_1(t; a_0, t_0) = a_0/\sqrt{(t+t_0)}$, $g_2(t; a_0, t_0) = a_0/(t+t_0)$, $g_3(t; a_0, \gamma) = a_0\gamma^t$, where t_0 is a predefined constant and $0 < \gamma < 1$.

Given the optimized S^*, following Hosseini and Sra (2015), we can translate S^* back to μ^* and Σ^* by $\mu^* = S^*[1:d, d+1]$, $\Sigma^* = S^*[1:d, 1:d] - S^*[1:d, d+1] * S^*[d+1, 1:d]/S^*[d+1, d+1]$

Algorithm 1: RAdam algorithm for GMM fitting

Input: learning rate α_w, a_0, t_0, momentum coefficients β_1, β_2, $\epsilon = 1e^{-6}$.
Output: estimate $\{w_k, S_k\}_{k=1}^K$ or $\{w_k, \mu_k, \Sigma_k\}_{k=1}^K$.
Initialize GMM parameter $\{w_{k,0}, S_{k,0}\}_{k=1}^K$, ensure $\sum_{k=1}^K w_{k,0} = 1$, $S_{k,0} \succ 0$.
Initialize natural gradients $\xi_{S_{k,0}} = \nabla^{\text{nat}}_{S_{k,0}} \mathcal{L}$ and the momentum, $M_{k,0} = \xi_{S_{k,0}}$, $v_{k,0} = ||\xi_{S_{k,0}}||_F^2$.

for $t = 0$ *to* T **do**
  for $k = 0$ *to* K **do**
    $w^-_{k,t+1} \leftarrow w_{k,t} + \alpha_w \nabla^{\text{nat}}_{w_{k,t}} \mathcal{L}$ ▷ update the weights
  end
  $w_{t+1} \leftarrow w^-_{t+1} / \sum_k (w^-_{k,t+1})$ ▷ normalize the weights
  $\alpha_{S,t} \leftarrow g(t; a_0, t_0)$ ▷ update the learning rate
  for $k = 0$ *to* K **do**
    $\xi_{S_{k,t}} \leftarrow \nabla^{\text{nat}}_{S_{k,t}} \mathcal{L}$ ▷ obtain the gradient on the tangent space
    $M^-_{k,t} \leftarrow \Gamma_{S_{k,t-1} \to S_{k,t}}(M_{k,t})$ ▷ vector transport
    $M_{k,t+1} \leftarrow \beta_1 M^-_{k,t} + (1-\beta_1)\xi_{S_{k,t}}$ ▷ update biased first moment estimate
    $v_{k,t+1} \leftarrow \beta_2 v_{k,t} + (1-\beta_2)||\xi_{S_{k,t}}||_F^2$ ▷ update biased second raw moment estimate
    $\hat{M}_{k,t+1} \leftarrow M_{k,t+1}/(1-\beta_1^t)$ ▷ biased-corrected first moment estimate
    $\hat{v}_{k,t+1} \leftarrow v_{k,t+1}/(1-\beta_2^t)$ ▷ biased-corrected second raw moment estimate
    $S_{k,t+1} \leftarrow R_{S_{k,t}}(\alpha_{S,t} \hat{M}_{k,t+1}/(\sqrt{\hat{v}_{k,t+1}} + \epsilon))$ ▷ retraction update
  end
end

4. Simulation study

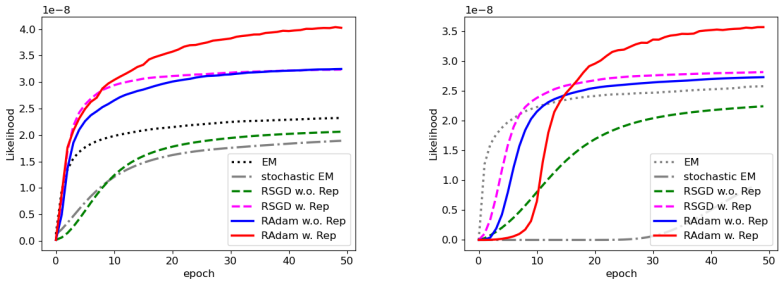
The simulated data is generated from a ground true GMM model, $\sum_{k=1}^K w_k \mathcal{N}(\cdot; \mu_k, \Sigma_k)$, where the dimension of data d, the number of components K, mean μ_k and covariance Σ_k are set according to the simulation study and we fix $w_k = 1/K$. In the following experiments, we simulated 4096 data points from this ground true GMM when the dimension $d \leq 100$, while 8192 data points for larger dimension cases.

It is well known that the performance of EM depends on the degree of separation of the mixture components (Xu and Jordan, 1996; Ma et al., 2000). To assess the impact of separation, we generate data of different degrees of separation with the following strategy (Verbeek et al., 2003): the distributions are sampled so that their means satisfy the inequality: $||\mu_i - \mu_j|| \geq c \max_{i,j}\{\text{tr}(\Sigma_i) - \text{tr}(\Sigma_j)\}, \forall_{i \neq j}$, where c models the degree of separation. We evaluate three levels of separation $c = 0.2$ (low), $c = 1$ (medium), and $c = 5$ (high).

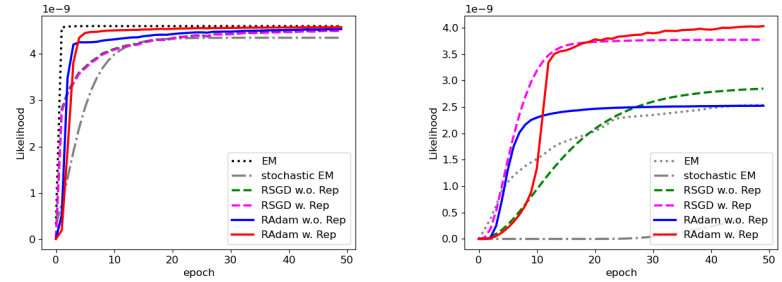
The proposed algorithms are implemented with the Pytorch package (Paszke et al., 2017). The default decreasing learning rate strategy for covariance matrice is $g_1(\cdot; 0.5, 10)$, while the learning rate for weights is fixed as 10^{-2}. For the momentum coefficients in RAdam algorithm, the default values are $\beta_1 = 1e^{-3}$ and $\beta_2 = 0.9$. The default batch size is 512 for data with dimension ($d \leq 100$) and 2048 for larger dimensions ($d = 200, 300$). An additional study on the selection of hyperparameters, including the learning rate, momentum coefficients, and batch size, can be found in the Appendix. In all experiments, we initialize the mixture parameters using both k-means and random initialization. We start all the methods by using the same initialization for a fair comparison. We terminate all methods with the same criteria: they stop either when the difference of average log-likelihood falls

below 10^{-6}, or when the number of epochs exceeds 50. We use the average log-likelihood as the metric for performance evaluation. For all experiments, we run the algorithms 10 times and report the mean and standard deviation of the average log-likelihood.

We first demonstrate the convergence of the proposed RSGD and RAdam compared with EM and stochastic EM. To compare the efficiency of reparameterization, we also implement the RSGD and RAdam algorithms for optimizing covariance matrices of a GMM without parameterization, denoted by 'RSGD w.o. Rep' and 'RAdam w.o. Rep'). Notice that to optimize the mean of this GMM, we use traditional SGD in Euclidean space. We take both low- and medium-level separation datasets of 50 dimensions as examples and report the log-likelihood with respect to epochs. As predicted by theory (Xu and Jordan, 1996; Ma et al., 2000), the EM converges slowly in the case of a low level of separation. As shown in Figure 1, the RSGD/RAdam with reparameterized GMM is superior to the RSGD/RAdam without reparameterization for both a better convergence rate and a higher final log-likelihood, which confirms the effectiveness of the reparameterization strategy. Compared to RSGD, RAdam has a favorable convergence rate, although in random initialization cases (Figure 1(b)subfigure and 1(d)subfigure), it takes some time to adjust the gradients to get better convergence. Moreover, the Radam algorithm always obtains a higher final log-likelihood value than the RSGD.



(a) initialized with k-means (b) initialized randomly



(c) initialized with k-means (d) initialized randomly

Figure 1: Comparison of convergence rate GMM fitting on a 50-dimensional dataset: (a),(b) low level of separation; (c) (d) medium level of separation.

We further evaluate the proposed method under different levels of separation (low, medium, high) and different dimensions $d = [5, 10, 40, 50, 100, 200, 300]$, with the number of components $K = 10$. Table 2 shows the average log-likelihood obtained by the RAdam, RSGD, and EM algorithms with both random and k-means initialization. For k-means initialization, the RAdam significantly outperforms EM when the level of separation is low

and medium, which is known to be challenging for EM. RSGD is inferior to RAdam in these two cases but still superior to EM. When the level of separation is high, which is known as an easy case for GMM fitting, the RAdam/RSGD obtains similar results with EM. For random initialization, both RAdam and RSGD outperform EM at all levels of separation. Due to the intrinsic stochastic optimization property, both RAdam and RSGD are less sensitive to local optima and more robust against random initialization. Moreover, this experiment confirms that our proposed methods are scalable to high-dimensional problems.

We here provide additional experiments to compare the computation complexity, the computation time, and the final obtained averaged log-likelihood (ALL). The algorithms for comparison are EM, RCG: Riemannian conjugate gradient (Hosseini and Sra, 2015), RLBFGS: Riemannian LBFGS (Hosseini and Sra, 2015), VTF-RLBFGS: Vector Transport Free Riemannian LBFGS (Godaz et al., 2021), our proposed RSGD and RAdam. As shown in Table 1, the proposed methods RSGD and RAdam both have favorable performance in computation time compared with RCG and VTF-RLBFGS. Meanwhile, our methods tend to obtain superior ALL for low and mid separation cases.

		EM	RCG	RLBFGS	VTF-RLBFGS	RSGD	RAdam
Low	Epoches	78	122	125	113	95	89
	Time(s)	10.186	23.595	105.586	39.660	39.880	61.390
	Time(s)/Epoch	0.131	0.193	0.845	0.351	0.420	0.690
	ALL	-17.568	-17.696	-17.639	-17.668	-17.175	-17.012
Mid	Epoches	32	61	123	43	45	49
	Time(s)	7.814	21.588	101.586	19.519	19.230	41.223
	Time(s)/Epoch	0.244	0.354	0.826	0.454	0.427	0.841
	ALL	-19.299	-19.219	-19.218	-19.217	-19.218	-19.197
High	Epoches	11	26	56	31	49	61
	Time(s)	2.797	20.875	80.611	30.211	23.340	42.730
	Time(s)/Epoch	0.254	0.803	1.439	0.975	0.476	0.700
	ALL	-14.511	-14.511	-14.511	-14.511	-14.512	-14.511

Table 1: Speed and log-likelihood comparisons for dimensions $d = 50$, number of components $K = 10$ and $N = 4096$ observations; results are reported on average of 10 run.

5. Conclusion

This work addresses the Riemannian SGD type algorithm for GMM fitting. The reparameterization strategy enables us to focus on the optimization of covariance matrices. We obtain the natural gradient updates by treating the reparameterized GMM on a statistical manifold. We propose the RAdam and RSGD algorithms, which allow us to optimize the augmented covariance matrices on a SPD matrix manifold. Extensive simulations are provided to verify that the RAdam and RSGD scale well to high-dimensional datasets and outperform the EM algorithm, particularly for difficult scenarios of GMM fitting.

Table 2: Comparison of average log-likelihood for datasets with different levels of separation and dimensions.

Separation	Method	$d = 5$	$d = 10$	$d = 40$	$d = 50$	$d = 100$	$d = 200$	$d = 300$
High	EM(random)	-4.01±0.04	-5.32±0.08	-14.26±0.23	-16.93±0.09	-29.76±1.29	-49.61±0.02	-49.87±0.02
	EM(kmeans)	***-3.65±0.02***	***-5.09±0.06***	**-12.27±0.00**	**-14.51±0.00**	**-11.40±0.00**	**-18.15±0.00**	**-9.72±0.00**
	RSGD(random)	-3.84±0.09	-5.48±0.13	-13.08±0.11	-15.54±0.09	-13.48±0.75	-27.68±0.61	-24.50±4.39
	RSGD(kmeans)	**-3.63±0.00**	**-5.03±0.00**	***-12.28±0.00***	***-14.59±0.00***	***-11.41±0.00***	**-18.15±0.00**	***-9.73±0.00***
	RAdam(random)	-3.94±0.11	-5.41±0.11	-12.73±0.09	-15.37±0.06	-16.05±0.12	-27.10±1.75	-24.55±1.19
	RAdam(kmeans)	**-3.63±0.00**	**-5.03±0.00**	**-12.27±0.00**	**-14.51±0.00**	**-11.40±0.00**	***-18.16±0.01***	-9.77±0.01
Medium	EM(random)	-3.34±0.00	-6.52±0.00	-16.79±0.03	-20.16±0.07	-22.65±0.07	-40.83±0.01	-45.28±0.04
	EM(kmeans)	***-3.23±0.00***	**-6.07±0.00**	-16.30±0.03	-19.33±0.04	-15.80±0.06	-27.46±0.14	-24.05±0.10
	RSGD(random)	***-3.23±0.00***	-6.08±0.01	-16.45±0.05	-19.48±0.04	-17.76±0.79	-33.31±0.74	-38.76±2.23
	RSGD(kmeans)	***-3.23±0.00***	**-6.07±0.00**	-16.26±0.01	***-19.23±0.02***	***-15.73±0.00***	***-27.10±0.02***	***-23.85±0.00***
	RAdam(random)	***-3.23±0.00***	***-6.08±0.00***	***-16.17±0.04***	-19.40±0.08	-16.00±0.15	-32.28±0.33	-29.52±1.51
	RAdam(kmeans)	**-3.22±0.00**	**-6.07±0.00**	**-16.24±0.00**	**-19.20±0.00**	**-15.71±0.00**	**-27.05±0.00**	**-23.78±0.01**
Low	EM(random)	-2.52±0.00	-4.95±0.00	-14.51±0.00	-17.37±0.01	-16.47±0.05	-32.90±0.02	-36.25±0.03
	EM(kmeans)	***-2.49±0.00***	-4.82±0.00	-14.64±0.01	-17.57±0.01	-15.93±0.03	-27.52±0.06	***-23.84±0.16***
	RSGD(random)	***-2.49±0.00***	-4.80±0.01	-14.37±0.06	-17.25±0.13	-15.93±0.79	-30.89±0.67	-33.83±0.99
	RSGD(kmeans)	***-2.49±0.00***	***-4.80±0.00***	-14.36±0.00	-17.18±0.01	-15.13±0.10	***-27.04±0.08***	-23.90±0.00
	RAdam(random)	***-2.49±0.00***	***-4.80±0.00***	***-14.33±0.04***	***-17.06±0.09***	***-14.59±0.25***	-30.01±0.63	-33.41±0.99
	RAdam(kmeans)	**-2.48±0.00**	**-4.79±0.00**	**-14.26±0.00**	**-17.02±0.01**	**-13.91±0.04**	**-27.04±0.02**	**-23.80±0.00**

Acknowledgments

This work was supported by National Key R&D Program of China No. 2021YFB3802103.

References

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Shun-ichi Amari. Neural learning in structured parameter spaces-natural riemannian gradient. *Advances in neural information processing systems*, pages 127–133, 1997.

Christian Beecks, Anca Maria Ivanescu, Steffen Kirchhoff, and Thomas Seidl. Modeling image similarity by Gaussian mixture models and the signature quadratic form distance. In *ICCV*, pages 1754–1761. IEEE, 2011.

Silvere Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

William M Campbell, Douglas E Sturim, and Douglas A Reynolds. Support vector machines using GMM supervectors for speaker verification. *IEEE signal processing letters*, 13(5): 308–311, 2006.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM - algorithm plus discussions on the paper. *Journal of the royal statistical society. Series B (methodological)*, page 1–38, 1997.

Reza Godaz, Benyamin Ghojogh, Reshad Hosseini, Reza Monsefi, Fakhri Karray, and Mark Crowley. Vector transport free Riemannian LBFGS for optimization on symmetric positive definite matrix manifolds. In *ACML*, pages 1–16. PMLR, 2021.

J. Hershey and P. Olsen. Approximating the Kullback Leibler divergence between Gaussian mixture models. *ICASSP*, 4:IV–317–IV–320, 2007.

Reshad Hosseini and Suvrit Sra. Matrix manifold optimization for gaussian mixtures. *Advances in Neural Information Processing Systems*, 28, 2015.

Chi Jin, Yuchen Zhang, Sivaraman Balakrishnan, M. Wainwright, and Michael I. Jordan. Local maxima in the likelihood of Gaussian mixture models: Structural results and algorithmic consequences. In *NIPS*, 2016.

Hiroyuki Kasai, Pratik Jawanpuria, and Bamdev Mishra. Riemannian adaptive stochastic gradient algorithms on matrix manifolds. In *ICML*, pages 3262–3271. PMLR, 2019.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint, arXiv:1412.698, 2014.

Wu Lin, Mohammad Emtiyaz Khan, and Mark Schmidt. Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. In *ICML*, pages 3992–4002. PMLR, 2019.

Wu Lin, Mark Schmidt, and Mohammad Emtiyaz Khan. Handling the positive-definite constraint in the Bayesian learning rule. In *ICML*, pages 6116–6126. PMLR, 2020.

Jinwen Ma, Lei Xu, and Michael I Jordan. Asymptotic convergence rate of the em algorithm for gaussian mixtures. *Neural Computation*, 12(12):2881–2907, 2000.

James Martens. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary Devito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Daniel Povey, Lukśš Burget, Mohit Agarwal, Pinar Akyazi, Kai Feng, Arnab Ghoshal, Ondřej Glembek, Nagendra Kumar Goel, Martin Karafiát, Ariya Rastrow, et al. Subspace Gaussian mixture models for speech recognition. In *ICASSP*, pages 4330–4333, 2010.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897. PMLR, 2015.

Lena Sembach, Jan Pablo Burgard, and Volker Schulz. A Riemannian newton trust-region method for fitting gaussian mixture models. *Statistics and Computing*, 32(1):8, 2022.

Minh-Ngoc Tran, Dang H Nguyen, and Duy Nguyen. Variational bayes on manifolds. *Statistics and Computing*, 31(6):1–17, 2021.

Jakob J Verbeek, Nikos Vlassis, and Ben Kröse. Efficient greedy learning of gaussian mixture models. *Neural computation*, 15(2):469–485, 2003.

Lei Xu and Michael I Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural Computation*, 8(1):129–151, 1996.

Appendix A. Derivation of the gradient for the reparameterized GMM

The reparameterized GMM is $\boldsymbol{q}_\phi(y) = \sum_{k=1}^K w_k q_\mathcal{N}(y; S_k$, where (w_k, S_k) (also denoted by ϕ) are the parameters to be optimized. Note that w_k are constrained to sum up to 1, $\sum_k w_k = 1$. Taking this into account by introducing the Lagrange multiplier, the gradient of $\mathcal{L}(\phi)$ with respect to each w_k (for $k = 1, ..., K-1$) can be derived as follows:

$$\begin{aligned}
\nabla_{w_k}\mathcal{L} &= \frac{\partial}{\partial w_k}\left[\frac{1}{N_s}\sum_{i=1}^{N_s}\log\sum_{j=1}^{K} w_j q_\mathcal{N}(y_i; S_j) + \lambda(\sum_{j=1}^{K} w_j - 1)\right] \\
&= \frac{1}{N_s}\sum_{i=1}^{N_s}\frac{\partial}{\partial w_k}\left[\log\sum_{j=1}^{K} w_j q_\mathcal{N}(y_i; S_j)\right] + \lambda \\
&= \frac{1}{N_s}\sum_{i=1}^{N_s}\frac{q_\mathcal{N}(y_i; S_k)}{\sum_{j'=1}^{K} w_{j'} q_\mathcal{N}(y_i; S_{j'})} + \lambda
\end{aligned} \tag{13}$$

To get the value of λ, we add all $\nabla_{w_k}\mathcal{L} * w_k$ (for $k = 1, ..., K$) together and the summation is set to 0, then it is easy to obtain $\lambda = -1$. So, the gradient for w_k is

$$\nabla_{w_k}\mathcal{L} = \frac{1}{N_s}\sum_{i=1}^{N_s}\frac{q_{\mathcal{N}}(y_i; S_k)}{\boldsymbol{q}_\phi(y_i)} - 1. \tag{14}$$

To optimize the parameter S_k in each component of the mixture, the gradient of $\mathcal{L}(\phi)$ with respect to S_k (for $k = 1, ..., K$) is derived as follows,

$$\begin{aligned}
\nabla_{S_k}\mathcal{L} &= \frac{\partial}{\partial S_k}\left[\frac{1}{N_s}\sum_{i=1}^{N_s}\log\sum_{j=1}^{K} w_j q_{\mathcal{N}}(y_i; S_j)\right] \\
&= \frac{1}{N_s}\sum_{i=1}^{N_s}\left[\frac{\partial}{\partial S_k}\log\sum_{j=1}^{K} w_j q_{\mathcal{N}}(y_i; S_j)\right] \\
&= \frac{1}{N_s}\sum_{i=1}^{N_s}\frac{w_k q_{\mathcal{N}}(y_i; S_k)\frac{1}{q_{\mathcal{N}}(y_i;S_k)}\frac{\partial}{\partial S_k}q_{\mathcal{N}}(y_i; S_k)}{\sum_{j'=1}^{K} w_{j'} q_{\mathcal{N}}(y_i; S_{j'})} \\
&= \frac{1}{N_s}\sum_{i=1}^{N_s}\frac{w_k q_{\mathcal{N}}(y_i; S_k)}{\boldsymbol{q}_\phi(y_i)}\frac{\partial}{\partial S_k}\log q_{\mathcal{N}}(y_i; S_k).
\end{aligned} \tag{15}$$

To obtain the gradient of $\log q_{\mathcal{N}}(y; S) := \log(2\pi\exp(\frac{1}{2})\mathcal{N}(y; S))$ w.r.t. S, we need the following derivatives w.r.t. a symmetric positive definite matrix S, $\frac{\partial \log|S|}{\partial S} = S^{-1}$ and $\frac{\partial \mathrm{trace}(S^{-1}yy^\top)}{\partial S} = -2Syy^\top S^{-1}$. Therefore,

$$\begin{aligned}
\frac{\partial \log q_{\mathcal{N}}(y|S)}{S} &= \frac{\partial \log(2\pi\exp(\frac{1}{2})\mathcal{N}(y; S))}{\partial} - \frac{\partial}{\partial S}\frac{1}{2}\left(d\log|2\pi| + \log|S| + y^\top S^{-1}y)\right) \\
&= -\frac{\partial}{\partial S}\frac{1}{2}\left(\log|S| + \mathrm{trace}(S^{-1}yy^\top)\right) = -\frac{1}{2}\left(S^{-1} - S^{-1}yy^\top S^{-1}\right)
\end{aligned} \tag{16}$$

Then, we can obtain the gradient of $\mathcal{L}(\phi)$ with respect to S_k (for $k = 1, ..., K$)

$$\nabla_{S_k}\mathcal{L} = \frac{1}{N_s}\sum_{i=1}^{N_s}\frac{w_k q_{\mathcal{N}}(y_i; S_k)}{\boldsymbol{q}_\phi(y_i)}\frac{1}{2}\left(S^{-1}y_i y_i^\top S^{-1} - S^{-1}\right) \tag{17}$$

Appendix B. Derivation of the natural gradient for the reparameterized GMM

To obtain the natural gradient, our aim is to minimize the loss function, meanwhile, subject to keeping the KL divergence within a constant c. Formally, it can be written as

$$\delta\phi^* = \arg\min_{\delta\phi}\mathcal{L}(\phi + \delta\phi) \;\text{ s.t. }\; KL[q_\phi||q_{\phi+\delta\phi}] = c \tag{18}$$

By a second-order Taylor relaxation, the KL divergence can be expressed as a function of FIM and the delta change in parameters between the two distributions, $KL[q_\phi||q_{\phi+\delta\phi}] \approx$

$\frac{1}{2}\delta\phi \cdot E_{q(\cdot;\phi)}[\nabla_\phi \log q(x;\phi)(\nabla_\phi \log q(x;\phi))^T] \cdot \delta\phi = \frac{1}{2}\delta\phi \cdot F_\phi \cdot \delta\phi$. Taking the Lagrangian relaxation of Eq.(18) and using the first-order Taylor relaxation of $\mathcal{L}(\phi + \delta\phi)$, we obtain

$$\begin{aligned} \delta\phi &= \underset{\delta\phi}{argmin}\left[\mathcal{L}(\phi+\delta\phi) + \lambda(KL[q_\phi||q_{\phi+\delta\phi}] - c)\right] && (19) \\ &\approx \underset{\delta\phi}{argmin}\left[\mathcal{L}(\phi) + \nabla\mathcal{L}(\phi)^T \cdot \delta\phi + \frac{1}{2}\lambda \cdot \delta\phi \cdot F_\phi \cdot \delta\phi - \lambda c\right] && (20) \end{aligned}$$

From now on, we use $\mathcal{L}$ and $\nabla\mathcal{L}$ instead of $\mathcal{L}(\phi)$ and$\nabla\mathcal{L}(\phi)$ for short. To minimize the function above, we set the derivative of Eq. (20) to zero that, $\nabla\mathcal{L} + \lambda \cdot F_\phi \cdot \delta\phi = 0$. By solving it for $\delta\phi$, we get $\delta\phi = -\frac{1}{\lambda}F_\phi^{-1}\nabla\mathcal{L}$. We can take the constant factor $-\frac{1}{\lambda}$ of relaxation into the learning rate. Then we have the *natural gradient* as

$$\nabla_\phi^{\text{nat}}\mathcal{L} = F^{-1}\nabla_\phi\mathcal{L} \tag{21}$$

Now, let us derive the natural gradient for the reparameterized GMM. Let $\phi = \{w_k, S_k\}_{k=1}^K$ denote the parameters of this GMM, that $q_\phi(x) = \sum_{k=1}^K w_k q_\mathcal{N}(\cdot|S_k)$ and $\mathbf{w} = [w_1, ..., w_k]^T$. We approximate $KL[q_\phi||q_{\phi+\delta\phi}]$ by its upper bound $KL[\mathbf{w}||\mathbf{w}+\delta\mathbf{w}]+\sum_{k=1}^K w_k KL[q_{S_k}||q_{S_k+\delta S_k}]$. As discussed above, the KL divergence between two continuous distributions can be approximated by FIM and the delta of the parameters that $\frac{1}{2}\delta\phi \cdot F_\phi \cdot \delta\phi$. By a second-order Taylor relaxation, the KL between two discrete variables $KL[\mathbf{w}||\mathbf{w}+\delta\mathbf{w}]$ can also be expressed as

$$KL[\mathbf{w}||\mathbf{w}+\delta\mathbf{w}] \approx -\sum_{k=1}^K w_k(\frac{1}{w_k}\delta w_k + \frac{-1}{2w_k^2} \cdot \delta w_k \cdot \delta w_k) = \frac{1}{2}\sum_{k=1}^K \frac{1}{w_k} \cdot \delta w_k \cdot \delta w_k \tag{22}$$

where since $\sum_{k=1}^K w_k = 1$, we have $\sum_{k=1}^K \delta w_k = 0$. By keeping this upper bound a constant c, the Lagrangian relaxation of the problem becomes

$$\begin{aligned} \delta\phi &= \underset{\delta\phi}{argmin}\left[\mathcal{L}(\phi+\delta\phi) + \lambda\left(KL[\mathbf{w}||\mathbf{w}+\delta\mathbf{w}] + \sum_{k=1}^K w_k KL[q_{S_k}||q_{S_k+\delta S_k}] - c\right)\right] && (23) \\ &\approx \underset{\delta\phi}{argmin}\left[\mathcal{L} + \nabla\mathcal{L}^T \cdot \delta\phi + \frac{\lambda}{2}\sum_{k=1}^K \frac{1}{w_k} \cdot \delta w_k \cdot \delta w_k + \frac{\lambda}{2}\sum_{k=1}^K w_k \cdot \delta S_k \cdot F_{S_k} \cdot \delta S_k - \lambda c\right] && (24) \end{aligned}$$

For the weight w_k of each mixture components, the derivative of the above function with respect to each w_k is $\nabla\mathcal{L} + \lambda\frac{1}{w_k} \cdot \delta w_k$. By setting it to zero and solving for δw_k, we get,

$$\delta w_k = -\frac{w_k}{\lambda}\nabla\mathcal{L} \tag{25}$$

For S_k in each mixture component, the derivative of the above function with respect to each S_k is $\nabla\mathcal{L} + \lambda \cdot w_k \cdot F_{S_k} \cdot \delta S_k$. By setting it to zero and solving for δS_k, we get,

$$\delta S_k = -\frac{1}{\lambda w_k}F_{S_k}^{-1}\nabla\mathcal{L} \tag{26}$$

Taking the constant factor $-\frac{1}{\lambda}$ of relaxation into the learning rate, the natural gradients for w_k and S_k in each component of the reparameterized GMM are

$$\begin{aligned} \nabla_{w_k}^{\text{nat}} &= w_k\nabla_{w_k}\mathcal{L} && (27) \\ \nabla_{S_k}^{\text{nat}} &= \frac{1}{w_k}2S_k\left(\nabla_{S_k}\mathcal{L}\right)S_k && (28) \end{aligned}$$

Appendix C. Operators on the SPD matrix manifold

Table 3: Operators on the SPD matrix manifold

Definition	Expression for the SPD manifold
Metric between ξ,η at S	$\rho_S(\xi,\eta) = tr(S^{-1}\xi S^{-1}\eta)$
Gradient at S if Euclidean gradient is $\nabla_E f$	$\nabla f(S) = \frac{1}{2}S\left(\nabla_E f(S) + [\nabla_E f(S)]^T\right)S$
Exponential map at S in direction ξ	$Exp_S(\xi) = S\exp(S^{-1}\xi)$
Parallel transport of ξ from S_1 to S_2	$\Gamma_{S_1,S_2}(\xi) = E\xi E^T$, $E = (S_2 S_1^{-1})^{1/2}$
Euclidean Retraction at S in direction ξ	$Ret_S(\xi) = S + \xi + \frac{1}{2}\xi S^{-1}\xi$
Frobenius norm of S, $\|\|S\|\|_F$	$tr(S^T S)^{1/2}$

Appendix D. The Riemannian SGD algorithm for GMM fitting

Algorithm 2: RSGD algorithm for GMM fitting

Input: learning rate α_w, a_0, t_0, momentum coefficients β_1, β_2, $\epsilon = 1e^{-6}$.
Output: estimate $\{w_k, S_k\}_{k=1}^K$ or $\{w_k, \mu_k, \Sigma_k\}_{k=1}^K$.
Initialize GMM parameter $\{w_{k,0}, S_{k,0}\}_{k=1}^K$, ensure $\sum_{k=1}^K w_{k,0} = 1$, $S_{k,0} \succ 0$.
for $t = 0$ *to* T **do**
  for $k = 0$ *to* K **do**
    $w^-_{k,t+1} \leftarrow w_{k,t} + \alpha_w \nabla^{\text{nat}}_{w_{k,t}} \mathcal{L}$ ▷ update the weights
  end
  $w_{t+1} \leftarrow w^-_{t+1} / \sum_k (w^-_{k,t+1})$ ▷ normalize the weights
  $\alpha_{S,t} \leftarrow g(t; a_0, t_0)$ ▷ update the learning rate
  for $k = 0$ *to* K **do**
    $\xi_{S_{k,t}} \leftarrow \nabla^{\text{nat}}_{S_{k,t}} \mathcal{L}$ ▷ obtain the gradient on the tangent space
    $S_{k,t+1} \leftarrow R_{S_{k,t}}(\alpha_{S,t}\xi_{S_{k,t}})$ ▷ retraction update
  end
end

Appendix E. Analysis of computational complexity

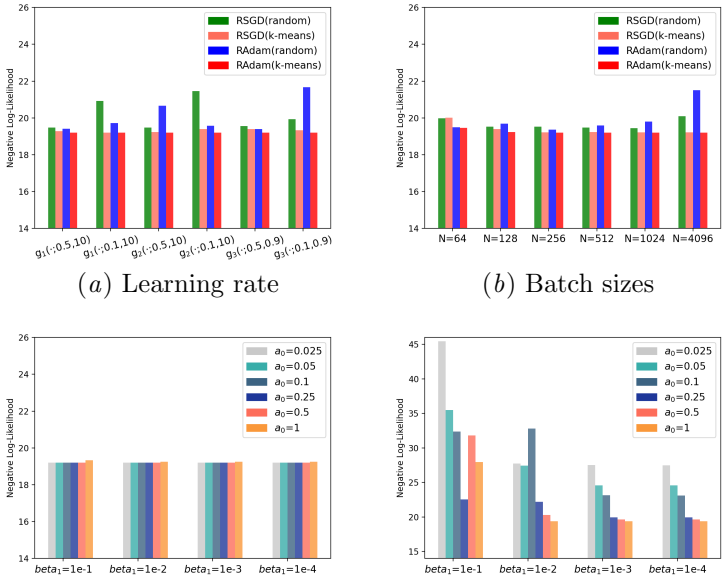
We compare the computational complexity between the EM algorithm and the proposed methods. For EM, the computational complexity of E-step and M-step are $O(Knd^2 + Kn^2d + Kd^3)$ and $O(Knd^2)$ respectively. Thus the overall computational complexity is $O(2Knd^2 + Kn^2d + Kd^3)$ (n is the number of samples, d is the dimension of data, K is the number of mixture components); For the proposed RAdam/RSGD, when using entire data in each iteration, the computation of natural gradient is actually a combination of calculating the assignment probability and the gradient for each component (similar to the

computation of M-step, that $O(2KnD^2 + Kn^2D + KD^3)$. The retraction and vector transport of the manifold involve matrix inversion and matrix multiplication, the computational complexity of manifold update is approximately $O(D^3)$, $(D = d+1)$. So, the overall computational complexity of the proposed algorithm in each epoch is $O(2KnD^2+Kn^2D+2KD^3)$; When using mini-batch data in each iteration, the computational complexity of the proposed algorithm becomes $O(2KnD^2 + Kn^2D/r + 2rKD^3)$, where $r = n/\text{batch-size}$. Through the comparison, we find that the complexity of the proposed algorithm is larger than EM, but not too much.

Appendix F. Study of the hyperparameter in the proposed algorithms

We evaluate the proposed algorithms against different hyperparameter settings: We compare the different learning rate decreasing strategies, such as $g_1(\cdot; 0.5, 10)$, $g_1(\cdot; 0.1, 10)$, $g_2(\cdot; 0.5, 10)$, $g_2(\cdot; 0.1, 10)$, $g_3(\cdot; 0.5, 0.9)$, $g_3(\cdot; 0.1, 0.9)$. The data are simulated with a medium level of separation with $K = 10$ and $d = 50$; We compare the model performance when using different batch sizes, $N_s = [64, 128, 256, 512, 1024, 4096]$. Figure 2(a) shows that the learning rate affects the performance of the algorithms. Generally, a larger initialization learning rate is preferred. Accordingly, the decreasing strategy $g_1(\cdot; 0.5, 10)$ is recommended. Figure 2(b) shows that the proposed method performs stably with different batch sizes. Note that small batch data help the proposed algorithms with random initialization. According to the comparison, a slightly large batch size, such as 512 is preferred.

We here also provide a study on the selection of momentum coefficients β_1 and the initial learning rate a_0 in the decreasing strategy $g_1(\cdot; a_0, 10)$. We set $\beta_2 = 0.9$, as by experiments it has no significant effects on the performance of RAdam. Figure 2 (c) and (d) show the performance of RAdam with different initial learning rates a_0 and different values of β_1. In the case of k-means initialization, the value for a_0 does not affect the performance much, while a smaller value is preferred. However, in the case of random initialization, a larger value of a_0 is preferred. So to balance between different initialization, we recommend the setting: $a_0 = 0.5$ and $\beta_1 = 1e-3$, $\beta_2 = 0.9$.



(a) Learning rate (b) Batch sizes

(c) RAdam with different a_0 and β_1 (k-means initialization) (d) RAdam with different a_0 and β_1 (random initialization)

Figure 2: The study of hyperparametes.