Robust Statistical Methods Using Student-t Distributions

A Thesis
Submitted in partial fulfillment of
the requirements for the degree of
Master of Technology
by

Kalpesh Patil
(130040019)

Supervisors:
Prof. S. N. Merchant
and
Prof. S. P. Awate

Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai 400076 (India)

9 June 2018

Dedicated to my parents

Acceptance Certificate

Department of Electrical Engineering

Indian Institute of Technology, Bombay

The thesis entitled "Robust Statistical Methods Using Student-t Distributions" submitted by Kalpesh Patil (130040019) may be accepted for being evaluated.

Date: 9 June 2018

Prof. S. N. Merchant

Approval Sheet

This thesis entitled "Robust Statistical Methods Using Student-t Distributions" by Kalpesh Patil is approved for the degree of Master of Technology.

____

____

____

Examiners

____

____

____

Supervisor (s)

____

Chairman

Date: ______________

Place: ______________

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this report. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Kalpesh Patil
(130040019)

Date: 9 June 2018

Abstract

Presence of large outliers in data usually affect the performance of any statistical method. It has been observed that statistical methods involving Student-t distributions are more robust to such outliers. We have exploited this property of Student-t distributions in several ways. First, we have developed a new generative model named Kernel-space Student-t Mixture Model (KSMM) and demonstrated its effectiveness in presence of large outliers while classifying non-linear data. We demonstrate results on simulated as well as real datasets. We also modify this model to incorporate sparsity (Sparse KSMM) and show its effectiveness on real datasets. We have also shown results on simulated data for an SMM learned on Kernel Hilbert sphere (KPGA-SMM). Lastly, we develop a technique to learn robust priors using Student-t Mixture Models (SMM). We have demonstrated results of this technique for reconstruction of simulated Functional MRI data.

Index terms: Robust outlier detection, kernel student-t mixture models, sparse models, EM, kernel tricks, principal geodesic analysis, robust reconstruction

Table of Contents

List of Figures

List of Tables

Chapter 1

Preliminaries

Mixture models have been successfully used to model multimodal distributions. In this chapter a brief introduction would be given for traditional Gaussian Mixture Models (GMM). Then we will point some of the limitations of GMM and describe how those can be overcome by Student-t Mixture Models (SMM).

1.1 Gaussian Mixture Models

GMM is one of the most widely used model for clustering. Bishop (2007, Chapter 9) has studied GMM in depth. Each data point is modeled as weighted sum of Gaussian distributions which are parameterized by $\{\mu_k, \Sigma_k, \pi_k\}$ (means, covariance matrices and weights for each component).

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k) \tag{1.1}$$

Cluster labels are treated as hidden variables in this model. The parameters are optimized using EM algorithm. EM algorithm consists of an *E-step* and and an *M-step*, which are performed alternatively. *E-step* computes expectation of log-likelihood using current estimates of parameters. *M-step* maximizes expectation found in *E-step*. This provides updates for the parameters of the model. Please refer (Bishop, 2007, Chapter 9) for detailed analysis of GMM using EM algorithm.

1.1.1 Limitations of GMM

While using GMM, an inherent assumption that datapoints from a particular cluster are generated from Gaussian distribution, is made. Although this is justifiable in the case of pure data corrupted with Gaussian noise, it fails to model large outliers. Gaussian distribution assigns very less probability for points far away from the center (outliers). Hence to model such outliers, a heavy tailed distribution in needed.

1.2 SMM

McLachlan *et al.* (2016) have studied finite mixture of Student-t distributions for unsupervised clustering. A student-t distribution can be looked upon as a combination of infinitely many Gaussian distributions with scaled covariance matrices. Define u as a scaling parameter for Σ for each data point. Assume u is generated from $Gamma(\frac{\nu}{2}, \frac{\nu}{2})$ distribution. Consider combination of such infinitely many scaled gaussians.

$$f(x;\mu,\Sigma,\nu) = \int_0^\infty \mathcal{N}\left(x_j;\mu,\frac{\Sigma}{u}\right)\Gamma\left(u;\frac{\nu}{2},\frac{\nu}{2}\right)du \tag{1.2}$$

Above integral results in student-t distribution given below

$$f(x_j;\mu,\Sigma,\nu) = \frac{\Gamma(\frac{\nu+p}{2})|\Sigma|^{-1/2}}{(\pi\nu)^{\frac{p}{2}}\Gamma(\frac{\nu}{2})(1+\delta(x_j;\mu,\Sigma)/\nu))^{\frac{\nu+p}{2}}} \tag{1.3}$$

$$\delta(x_j;\mu,\Sigma) = (x_j-\mu)^T\Sigma^{-1}(x_j-\mu) \tag{1.4}$$

Here p is dimensionality of the data points and ν is the parameter for degree of freedom. It controls the heavy-tailedness of the distribution. Lower DoF implies heavier tails. At $\nu = \infty$ it approaches Gaussian distribution. In SMM we have two sets hidden variables, class labels and scaling factors. Label(z_{ij}) is 1 if x_j belongs to class i, else 0. According to the definition of the 'scaling parameter' u_i, we can define following random variables.

$$X_j \mid (u_j, z_{ij}=1) \sim \mathcal{N}\left(\mu_i, \frac{\Sigma_i}{u_j}\right) \tag{1.5}$$

$$U_j \mid (z_{ij}=1) \sim \Gamma\left(\frac{v_i}{2}, \frac{v_i}{2}\right) \tag{1.6}$$

The iterative updates obtained using EM for the above formulation of SMM are described in McLachlan *et al.* (2016), which are mentioned below.

$$\pi_l^{t+1} = \frac{1}{N}\sum_{i=1}^{N}\tau_{li}^t \tag{1.7}$$

$$\mu_l^{t+1} = \frac{\sum_{i=1}^N w_{li}^t\tau_{li}^t x_i}{\sum_{i=1}^N w_{li}^t\tau_{li}^t} \tag{1.8}$$

$$\Sigma_l^{t+1} = \frac{\sum_{i=1}^N w_{li}^t\tau_{li}^t(x_i-\mu_l)(x_i-\mu_l)^T}{\sum_{i=1}^N \tau_{li}^t} \tag{1.9}$$

In above,

$$\tau_{li}^{t} = \frac{\pi_l f(x_i; \mu_l^{t-1}, \Sigma_l^{t-1}, \nu_l^{t-1})}{f(x_i; \theta^{t-1}})\tag{1.10}$$

$$w_{li}^{t} = \frac{\nu_l^t + p}{\nu_l^t + (x_i - \mu_l^t)^T (\Sigma_l^t)^{-1} (x_i - \mu_l^t)}\tag{1.11}$$

The updates for $\{\nu_l\}$ don't have closed form solution. They are obtained after solving the eqn. 17 in McLachlan *et al.* (2016). Further Hennig (2004) have shown that outliers have to be much larger for the breakdown of an SMM compared to that of GMM.

Chapter 2

Kernel Student-t Mixture Model

In this chapter, we will discuss about the novel generative model formulated by us. Initially a brief introduction to Kernel Space will be given along with major prior arts in this domain. Then we will give mathematical formulation of our model. Finally we will show effectiveness of our model in classifying data having large outliers on a simulated as well as real dataset.

2.1 Prior Arts

The basic idea behind Kernel Spaces is to be able to map input data from low dimensional space to a higher dimensional subspace, also called as Kernel Space. The higher dimensional subspace may also be infinite dimensional. Such mapping provides higher separability in high dimensional subspace. For example, non-linearly separated data in euclidean space may become linearly separated in Kernel space, which makes it easier for classification.

$$x \mapsto \Phi(x) \tag{2.1}$$

$$G_{i,j} = \langle x_i, x_j \rangle_{\mathcal{H}} \tag{2.2}$$

But the biggest problem with such mapping is that it is explicitly computing $\Phi(x)$ is computationally expensive. Also sometimes, its not even possible to compute such features explicitly. Hence to get rid of the explicit feature computation, kernel tricks are employed. It is possible to compute dot product (inner product) of feature vectors in kernel space, without explicitly computing them. Kernel Tricks allow us to carry out required task with the help only the dot products computed earlier. Kernel trick is very popular in literature for dealing with non-linear data. A Kernel matrix, also known as Gram Matrix ($G_{i,j}$) is supposed to follow certain properties like symmetry, positive semi-definiteness etc.

Kernel tricks have traditionally been very popular in Support Vector Machine (KSVM)(Boser *et al.*, 1992). The major contribution was to introduce kernel tricks for finding optimal separating hyperplane in kernel space. Later these kernel tricks were used in various scenarios. Kernel PCA(Schölkopf *et al.*, 1997) formulated usual Principal Component Analysis in kernel space. The major contribution of this paper was to prove a relation of eigenvalues and eigenvectors of covariance matrix with Gram kernel matrix. Kernel GMM (Wang *et al.*, 2003) incorporated kernel tricks for Gaussian Mixture Models. They devised a way to find updates of EM algorithm used in GMM's parameter estimation using dot products only.

2.2 Kernel Tricks for Student-t Mixture Model

Let X be a random variable taking values x in *input space* $\mathcal{X}$. Let $\{x_n\}_{n=1}^{N}$ be a set of observations in input space. Let $G(\cdot,\cdot)$ be a real-valued Mercer kernel with an associated map $\Phi(\cdot)$ that maps x to $\Phi(x) := G(\cdot, x)$ in a RKHS $\mathcal{H}$ (Aronszajn, 1950; Scholkopf and Smola, 2002). Consider two points in RKHS, within the span of the mapped data, represented as $f := \sum_{i=1}^{I} \alpha_i \Phi(x_i)$ and $f' := \sum_{j=1}^{J} \beta_j \Phi(x_j)$ where the weights $\alpha_i \in \mathbb{R}$ and $\beta_j \in \mathbb{R}$. The inner product $\langle f, f' \rangle_{\mathcal{H}} := \sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_i \beta_j G(x_i, x_j)$. The norm $\|f\|_{\mathcal{H}} := \sqrt{\langle f, f \rangle_{\mathcal{H}}}$. When $f, f' \in F \backslash \{0\}$, let $f \otimes f'$ be the rank-one operator defined as $f \otimes f'(h) := \langle f', h \rangle_{\mathcal{H}} f$.

Let $Y := \Phi(X)$ be the random variable taking values y in RKHS. In Wang *et al.* (2003), Y is assumed to be distributed according a Gaussian Mixture Model. Here we assume that Y is distributed according to a Student-t Mixture Model. We show that the above mentioned EM updates for SMM can be computed using kernel tricks. Let's define parameters $\{a_{li}\}$ and $\{b_{li}\}$ such that

$$\mu_l = \sum_{i=1}^{N} a_{li}^2 \phi(x_i) \tag{2.3}$$

$$\Sigma_l = \sum_{i=1}^{N} b_{li}^2 (\phi(x_i) - \mu_l) \otimes (\phi(x_i) - \mu_l). \tag{2.4}$$

2.3 EM Algorithm for KSMM

Consider a Student-t Mixture Model with g components in Kernel space as follows.

$$f(\phi(x_i); \theta) = \sum_{l=1}^{g} \pi_l f(\phi(x_i); a_l, b_l, \nu_l) \tag{2.5}$$

Here $\theta = \{a_1, a_2 \cdots a_g, b_1, b_2 \cdots b_g, \nu_1, \nu_2 \cdots \nu_g, \pi_1, \pi_2 \cdots \pi_g\}$ is set of all parameters. The complete data vector is given by $y_c = \{y_1, y_2 \cdots y_n, z_1, z_2 \cdots z_n, u_1, u_2 \cdots u_n\}$. In this

$\{y_1, y_2 \cdots y_n\}$ denote the observed data, $\{z_1, z_2 \cdots z_n\}$ denote the component label vectors such that $z_{il} = 1$, if y_i belongs to component l. $\{u_1, u_2 \cdots u_n\}$ are scaling parameters as described in an earlier section. The complete likelihood ($L_c(\theta)$) can be calculated as follows.

$$\log(L_c(\theta)) = \sum_{j=1}^{n} \sum_{l=1}^{g} z_{jl} \log(Pr(y_j, z_{jl}, u_j)) \tag{2.6}$$

$$= \sum_{j=1}^{n} \sum_{l=1}^{g} z_{jl} \log(Pr(y_j | z_{jl}, u_j)) + z_{jl} \log(Pr(u_j | z_{jl})) + z_{jl} \log(Pr(z_{jl})) \tag{2.7}$$

$$= \sum_{l=1}^{g} \sum_{j=1}^{n} z_{jl} \log(\pi_l) + z_{jl} \log\Big(\Gamma\Big(u_j; \frac{v_l}{2}, \frac{v_l}{2}\Big)\Big) + z_{jl} \log\Big(\mathcal{N}\Big(y_j; \mu_l, \frac{\Sigma_l}{u_j}\Big)\Big) \tag{2.8}$$

$$= \sum_{l=1}^{g} \sum_{j=1}^{n} z_{jl} \log(\pi_l) + z_{jl}\Big(- \log\Gamma\Big(\frac{\nu_l}{2}\Big) + \frac{\nu_l}{2} \log\frac{\nu_l}{2} + \frac{1}{2}\nu_l(\log u_j - u_j) - \log u_j\Big)$$

$$+ z_{jl}\Big(\frac{1}{2} p \log(2\pi) - \frac{1}{2}\log|\Sigma_l| - \frac{1}{2} u_j \delta_{lj}\Big) \tag{2.9}$$

2.3.1 E-step

We have two sets of hidden variables which are z_{jl} and u_j. If we observe expression of $L_c(\theta)$ we get three quantities of hidden variables, which are z_{jl}, u_j and $\log(u_j)$. Note that these three are the only terms which will contain random variable of the posterior distribution, rest of the terms will be constant w.r.t. to that variable. Hence expectation w.r.t. posterior needs to be calculated for these three terms only.

- $E_{\theta^t}(Z_{jl}|Y_j)$

$$E_{\theta^t}(Z_{jl}|Y_j) = \tau_{lj}^t = \frac{\pi_l^t f(y_j; \mu_l^t; \Sigma_l^t; \nu_l^t)}{f(y_j; \theta^t)} \tag{2.10}$$

- $E_{\theta^t}(U_j|Y_j, Z_{jl})$

  We know that Γ distribution is the conjugate prior distribution for U_j. Therefore posterior distribution of U_j will also be Γ with different parameters. It can be proved that

$$U_j|(y_j, z_{jl} = 1) \sim \Gamma(m_{1l}, m_{2l}) \tag{2.11}$$

$$m_{1l} = \frac{1}{2}(v_l + p) \tag{2.12}$$

$$m_{2l} = \frac{1}{2}(v_l + \delta_{lj}) \tag{2.13}$$

$$E_{\theta^t}(U_j|(y_j, z_{lj} = 1) = \frac{m_{1i}}{m_{2i}} = \frac{\nu_l + p}{\nu_l + \delta_{lj}} = w_{lj} \tag{2.14}$$

- $E_{\theta^t}(\log(U_j)|Y_j, Z_{lj})$

  If R is distributed according to $\Gamma(\alpha,\beta)$, then $E[logR] = \psi(\alpha) - \log\beta$, where $\psi(\alpha)$ is a 'Diagamma function'.

$$E_{\theta^t}(\log(U_j)|y_j, z_{lj} = 1) = \psi(m_{1l}) - \log(m_{2l}) \tag{2.15}$$

After substituting these expressions in expectation of the likelihood function, we obtain following relation.

$$Q(\theta;\theta^t) = Q_1(\{\pi_l\};\theta^t) + Q_2(\{\nu_l\};\theta^t) + Q_3(\{\mu_l\},\{\Sigma_l\};\theta^t) \tag{2.16}$$

$$Q_1 = \sum_{l=1}^{g}\sum_{j=1}^{n} \tau_{lj}^t \log(\pi_l) \tag{2.17}$$

$$Q_2 = \sum_{l=1}^{g}\sum_{j=1}^{n} \tau_{lj}^t Q_{2lj}(\nu_l;\theta^t) \tag{2.18}$$

$$Q_3 = \sum_{l=1}^{g}\sum_{j=1}^{n} \tau_{lj}^t Q_{3lj}(\mu_l,\Sigma_l;\theta^t) \tag{2.19}$$

After ignoring terms not involving ν_i in $Q_{2j}(\nu_i;\Psi^k)$ and ξ_i in $Q_{3j}(\theta_i;\Psi^k)$ we obtain final expressions as follows.

$$Q_{2lj}(\nu_l;\theta^t) = -\log\Big(\Gamma(\frac{\nu_l}{2})\Big) + \frac{1}{2}\nu_l\log(\nu_l) + \frac{1}{2}\nu_l\Big\{\sum_{j=1}^{n}(\log(u_{lj}^t) - u_{lj}^t) + \psi\Big(\frac{\nu_l^t + p}{2}\Big) - \log\Big(\frac{\nu_l^t + p}{2}\Big)\Big\}$$

$$Q_{3lj}(\mu_l,\Sigma_l;\theta^t) = -\frac{1}{2}p\log(2\pi) - \frac{1}{2}\log(|\Sigma_l|) + p\log(w_{lj}^t) - \frac{1}{2}w_{lj}\delta_{lj}$$

2.3.2 M-step

Maximizing Q function to obtain updates for parameters follows. Updates for $\{\pi_l\}$ requires us to consider only Q_1 and the updates are given below.

$$\{\pi^{t+1}\} = \max_{\pi} Q_1(\{\pi_l\};\theta^t) \text{ s.t. } \sum_{l=1}^{g}\pi_l = 1 \tag{2.20}$$

$$\pi_l^{t+1} = \frac{1}{n}\sum_{j=1}^{n}\tau_{lj}^t \tag{2.21}$$

To obtain updates for $\{\mu_l\}$ and $\{\Sigma_l\}$ and effectively $\{a_{li}\}$ and $\{b_{li}\}$, we only need to consider Q_3. For a given l, $\sum_{j=1}^{n} Q_{3jl}$ can be considered as log-likelihood function formed by n independent observations $y_1, y_2 \cdots y_n$ with common mean μ_l and different covariance matrices $\frac{\Sigma_l}{w_{l1}^t}, \frac{\Sigma_l}{w_{l2}^t} \cdots \frac{\Sigma_l}{w_{ln}^t}$ respectively. This results in following updates.

$$\mu_l^{t+1} = \frac{\sum_{j=1}^{n}\tau_{lj}^t w_{lj}^t y_j}{\sum_{j=1}^{n}\tau_{lj}^t w_{lj}^t} \tag{2.22}$$

But,

$$\mu_l^{t+1} = \sum_{j=1}^{n} (a_{lj}^{t+1})^2 y_j \tag{2.23}$$

$$\therefore a_{lj}^{t+1} = \sqrt{\frac{\tau_{lj}^t w_{lj}^t}{\sum_{j=1}^n \tau_{lj}^t w_{lj}^t}} \tag{2.24}$$

Similarly for Σ_l we obtain,

$$\Sigma_l^{t+1} = \frac{\sum_{j=1}^n \tau_{lj}^t w_{lj}^t (y_j - \mu_l^{t+1}) \otimes (y_j - \mu_l^{t+1})}{\sum_{j=1}^n \tau_{lj}^t} \tag{2.25}$$

But,

$$\Sigma_l^{t+1} = \sum_{j=1}^{n} (b_{lj}^{t+1})^2 (y_j - \mu_l^{t+1}) \otimes (y_j - \mu_l^{t+1}) \tag{2.26}$$

$$\therefore b_{lj}^{t+1} = \sqrt{\frac{\tau_{lj}^t w_{lj}^t}{\sum_{j=1}^n \tau_{lj}^t}} \tag{2.27}$$

Since the updates for $\{\nu_l\}$ are not available in closed form and they involve solving complicated equations, we treat them as tunable hyperparameters.

2.4 Eigen Analysis of Covariance

Let's define $\tilde{\Phi}_l(x_j) = \Phi(x_j) - \mu_l$. Let's define centered kernel matrix ($\tilde{\mathcal{K}}_l$) for class l as follows.

$$\tilde{\mathcal{K}}_l(i,j) = \langle b_{li}\tilde{\Phi}_l(x_i), b_{lj}\tilde{\Phi}_l(x_j)\rangle_{\mathcal{H}} \tag{2.28}$$

$$= b_{li}b_{lj}\Big(G(i,j) - \sum_{m=1}^{N} a_{lm}^2 G(i,m) - \sum_{m=1}^{N} a_{lm}^2 G(m,j) + \sum_{m=1}^{N}\sum_{n=1}^{N} a_{lm}^2 a_{ln}^2 G(m,n)\Big) \tag{2.29}$$

$\tilde{K}$ can be computed using G and parameters $\{a_{li}\}$ and $\{b_{li}\}$. Let v be the eigenfunction of Σ_l and λ be corresponding eigenvalue.

$$v = \frac{\Sigma_l v}{\lambda} = \frac{1}{\lambda}\sum_{j=1}^{n} b_{lj}^2 \tilde{\Phi}_l(x_j) \otimes \tilde{\Phi}_l(x_j) v \tag{2.30}$$

$$= \frac{1}{\lambda}\sum_{j=1}^{n} b_{lj}^2 \langle \tilde{\Phi}_l(x_j), v\rangle_{\mathcal{H}} \tilde{\Phi}_l(x_j) \tag{2.31}$$

$$= \sum_{j=1}^{n} b_{lj}\, \beta_{lj} \tilde{\Phi}_l(x_j) \tag{2.32}$$

For any k in $\{1, 2 \cdots n\}$,

$$\langle b_{lk}\tilde{\Phi}_l(x_k), \Sigma_l v\rangle_{\mathcal{H}} = \lambda\langle b_{lk}\tilde{\Phi}_l(x_k), v\rangle_{\mathcal{H}} \tag{2.33}$$

$$\langle b_{lk}\tilde{\Phi}_l(x_k), \sum_{j=1}^{n} b_{lj}^2\tilde{\Phi}_l(x_j) \otimes \tilde{\Phi}_l(x_j) \sum_{i=1}^{n} b_{li}\beta_{li}\tilde{\Phi}_l(x_i)\rangle_{\mathcal{H}} = \lambda\langle b_{lk}, \tilde{\Phi}_l(x_k), \sum_{i=1}^{n} b_{li}\beta_{li}\tilde{\Phi}_l(x_i)\rangle_{\mathcal{H}} \tag{2.34}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\langle b_{lk}\tilde{\Phi}_l(x_k), b_{lj}\tilde{\Phi}_l(x_j)\rangle_{\mathcal{H}}\langle b_{lj}\tilde{\Phi}_l(x_j), b_{li}\tilde{\Phi}_l(x_i)\rangle_{\mathcal{H}}\beta_{li} = \lambda\sum_{i=1}^{n}\langle b_{lk}\tilde{\Phi}_l(x_k), b_{li}\tilde{\Phi}_l(x_i)\rangle_{\mathcal{H}}\beta_{li} \tag{2.35}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\tilde{\mathcal{K}}_l(k, j)\tilde{\mathcal{K}}_l(j, i)\beta_{li} = \lambda\sum_{i=1}^{n}\tilde{\mathcal{K}}_l(k, i)\beta_{li} \tag{2.36}$$

Since this is true for all $k \in \{1, 2, \cdots n\}$, stacking up these scalars together, we obtain

$$\tilde{\mathcal{K}}_l^2\beta_l = \lambda\tilde{\mathcal{K}}_l\beta_l \tag{2.37}$$

$$\tilde{\mathcal{K}}_l\beta_l = \lambda\beta_l \tag{2.38}$$

Thus, we proved a relation between eigenfunction of Σ_l and its eigenvalue with eigenvector of matrix $\tilde{\mathcal{K}}_l$ and corresponding eigenvalues.

Computation of Probability function

The student-t probability distribution which needs to be computed for each data point is given below.

$$f(\phi(x_i); \mu_l, \Sigma_l, \nu_l) = \frac{\Gamma(\frac{\nu_l+p}{2})|\Sigma_l|^{-1/2}}{(\pi\nu_l)^{\frac{p}{2}}\Gamma(\frac{\nu_l}{2})(1 + \frac{\delta_{li}}{\nu_l}))^{\frac{\nu_l+p}{2}}} \tag{2.39}$$

Here we need to compute $|\Sigma_l|$ and δ_{lj} to find the probability. Those values can be computed as follows.

$$\Sigma_l = \sum_{m=1}^{n}\lambda_{lm}v_{lm} \otimes v_{lm} \tag{2.40}$$

$$\Sigma_l^{-1} = \sum_{m=1}^{n}\frac{v_{lm} \otimes v_{lm}}{\lambda_{lm}} \tag{2.41}$$

$$\delta_{li} = \langle(\phi(x_i) - \mu_l), \Sigma_l^{-1}(\phi(x_i) - \mu_l)\rangle_{\mathcal{H}} \tag{2.42}$$

$$= \langle\tilde{\phi}_l(x_i), \sum_{m=1}^{n}\frac{v_{lm} \otimes v_{lm}}{\lambda}\tilde{\phi}_l(x_i)\rangle_{\mathcal{H}} \tag{2.43}$$

$$= \sum_{m=1}^{n}\frac{\langle v_{lm}, \tilde{\phi}_l(x_i)\rangle_{\mathcal{H}}^2}{\lambda_m} \tag{2.44}$$

$$\langle v_{lm}, \tilde{\phi}_l(x_i)\rangle_{\mathcal{H}} = \sum_{j=1}^{n}\beta_{lmj}b_{lj}\langle\tilde{\phi}_l(x_j), \tilde{\phi}_l(x_i)\rangle_{\mathcal{H}} = \sum_{j=1}^{n}\beta_{lmj}b_{lj}G(j, i) \tag{2.45}$$

Also,

$$|\Sigma_l| = \prod_{k=1}^{N} \lambda_{lk} \tag{2.46}$$

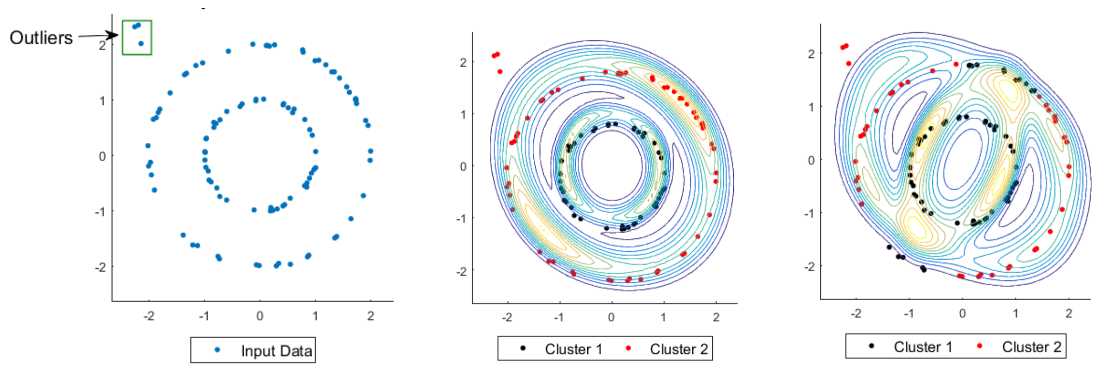
Thus probabilities can be computed only using inner products

2.5 KSMM Algorithm

Algorithm 1 summarizes the iterative EM based technique for parameter estimation described so far. The algorithm terminates either at convergence or after maximum number of iterations.

```
Input: A set of points $\{x_n\}_{n=1}^N$ in input space. Gram matrix $G$ underlying a kernel
  such that $G(i, j) = \langle\phi(x_i), \phi(x_j)\rangle_{\mathcal{H}}$. Number of clusters $g$, degree of freedom
  parameters $\{\nu_l\}$ . Set iteration number $t = 0$.
Initialization;
  1. Initialize $\tau_{li}$ such that $\sum_{l=1}^{g} \tau_{li} = 1 \ \forall \ i$
  2. Initialize $w_{li}$. It is better to initialize all $w_{li}$ with 1, because it would initialize like a
     KGMM
while stopping criterion == false do
    1. Compute $\{\pi_l\}$, $\{a_{li}\}$ and $\{b_{li}\}$ from the updates mentioned earlier using old values of
       $\{\tau_{li}\}$ and $\{w_{li}\}$.
    2. Compute matrices $\{\tilde{\mathcal{K}}_l\}$ and their eigenvectors and eigenvalues
    3. Compute $\{\delta_{li}\}$ using the the eigenvectors and eigenvalues of $\{\tilde{\mathcal{K}}_l\}$. to further
       calculate $Pr(\phi(x_i); \mu_l, \Sigma_l, \nu_l)$.
    4. Update $\{\tau_{li}\}$ and $\{w_{li}\}$
    5. Check stopping criterion for convergence i.e. either $t > t_{max}$ or
       $\sum_{i=1}^{n} \sum_{l=1}^{g} (\tau_{li}^{t} - \tau_{li}^{t-1})^2 < \epsilon$.
end
Output: A set of optimal parameters $\{\pi_l\}$, $\{a_{li}\}$ and $\{b_{li}\}$ representing the
student-t mixture model in kernel space.
```

Algorithm 1: KSMM algorithm



(a) Synthetic 2D data (b) KSMM (c) KGMM

Figure 2.1: KSMM vs KGMM on synthetic 2D dataset containing outliers

2.6 Experimental Results

2.6.1 Synthetic Dataset

Here we demonstrate effectiveness of our method on a task of unsupervised clustering on a synthetic dataset. This experiment is similar to the first experiment mentioned in Wang *et al.* (2003). Input data consists of 200 points along two concentric circles. It is very clear that normal euclidean space methods like GMM or SMM can't be used to model this data, since it is non-ellipsoidal. One needs to use kernel based methods to model this data. The effectiveness of KGMM against usual GMM was shown in Wang *et al.* (2003). We show that KSMM performs better than KGMM in presence of outliers.

We used polynomial kernel of degree 2 ($\langle\phi(x_i),\phi(x_j)\rangle_{\mathcal{H}} = (x_i^T x_j + c)^2$). In Figure 2.1 shows results of this experiment. Figure 4.1a shows input data points with outliers added in the top left corner. Figure 3.5d and Figure 2.1c show results of KSMM and KGMM respectively. Point with highest probability belonging to the cluster has been assigned with that cluster color. Probability contours are also plotted. It can easily be observed that KSMM correctly assigns cluster labels, while KGMM does not. Also probability contours of KSMM are still more or less circular and not much affected by the outliers. But probability contours of KGMM are significantly affected. Hence, KSMM is proved to be performing better than KGMM in presence of outliers. Thus, KSMM shows robustness against such outliers and correctly models the data even in presence of outliers, while KGMM breaks down completely and is not able to model the data.

2.6.2 Real Datasets

Here we try to tackle the problem of robust unsupervised outlier detection. We show that KSMM performs better than other state of the art methods when there are large number of outliers present in the training data. We benchmark our methods against other unsupervised outlier detection methods like One Class Support Vector Machine (Manevitz and Yousef, 2001), Support Vector Data Description (Tax and Duin, 2004) and Kernel Gaussian Mixture Models (Wang *et al.*, 2003). We have shown results on MNIST (LeCun *et al.*, 2010), ORL (Samaria and Harter, 1994), Imagenet (Deng *et al.*, 2009), Breast Cancer (Wolberg and Mangasarian, 1990) and Ionosphere (Sigillito *et al.*, 1989).

Table 2.1: Datasets Description

Dataset	Method of feature construction	Inlier class	No. of components (g)
MNIST	Vectorizing raw pixel values of images	images of digit "1"	4
ORL	First 30 PCA components of images	images of 30 subjects	4
Imagenet	2 PCA components of features extracted from fc1 layer of VGG-16	Images of class 'flower'	4
Breast cancer	provided numerical features	provided ('benign' data)	2
Ionosphere	provided numerical features	provided ('bad' radar data)	2

For MNIST, we use images of digit "1" as inliers and images of other digits as outliers. Raw pixel values of images after vectorization are taken as features. We train our model with 800 inlier with varying number of outliers in training images. An inlier class is modeled using KSMM/KGMM with 4 components ($g = 4$). The test dataset consist of equal number of inlier and outlier images. The test points whose probability value is below certain threshold are declared as outliers. Finally, accuracy on test dataset is considered as performance metric. Similar methodology is followed for all other dataset, with differences in method of constructing feature vectors and number of mixture components to model inlier class. For ORL dataset, we use images of faces of a subset of 30 subjects as inliers and remaining subjects as outliers. We have used first 30 PCA coefficients as feature vectors. For Imagenet, we considered 800 images of the class 'flower' as inliers and those of others as outliers. We compute 4096 dimensional output for each image obtained at fc1 layer of pretrained VGG-16 (Simonyan and Zisserman, 2014). Later we take first few PCA coefficients as feature vectors. Inlier class for both ORL and Imagenet are modeled using 4 components ($g = 4$). For Breast Cancer and Ionosphere datasets, outliers are provided by ODDS (Rayana, 2016). In breast cancer dataset, the data points belonging to malignant class are considered as outliers. In Ionosphere dataset, the data points belonging to 'bad' radar returns are considered as outliers. For both of these datasets, inlier class is modeled using 2 components ($g = 2$). Table 2.1 summarizes the construction of datasets, inlier class, outlier class etc. For all datasets we use the popular Gaussian kernel $\kappa(x_i, x_j) := \exp(-0.5\|x_i - x_j\|_2^2/\sigma^2)$ with appropriately tuned σ. The experiment is repeated multiple with times with variations in training dataset, test dataset and outliers to obtain box plots of the performance of each of these methods. It can be observed that KSMM outperforms all other methods when there are large number of outliers are present in the training data and hence turn out to be a robust method.

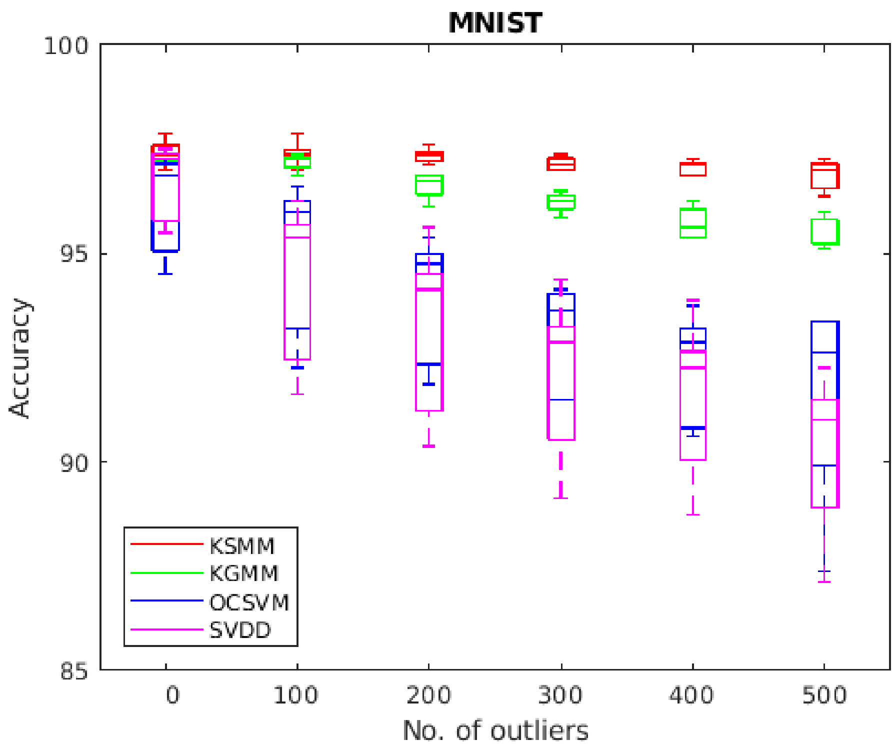


Figure 2.2: Accuracy with varying number of outliers for MNIST dataset

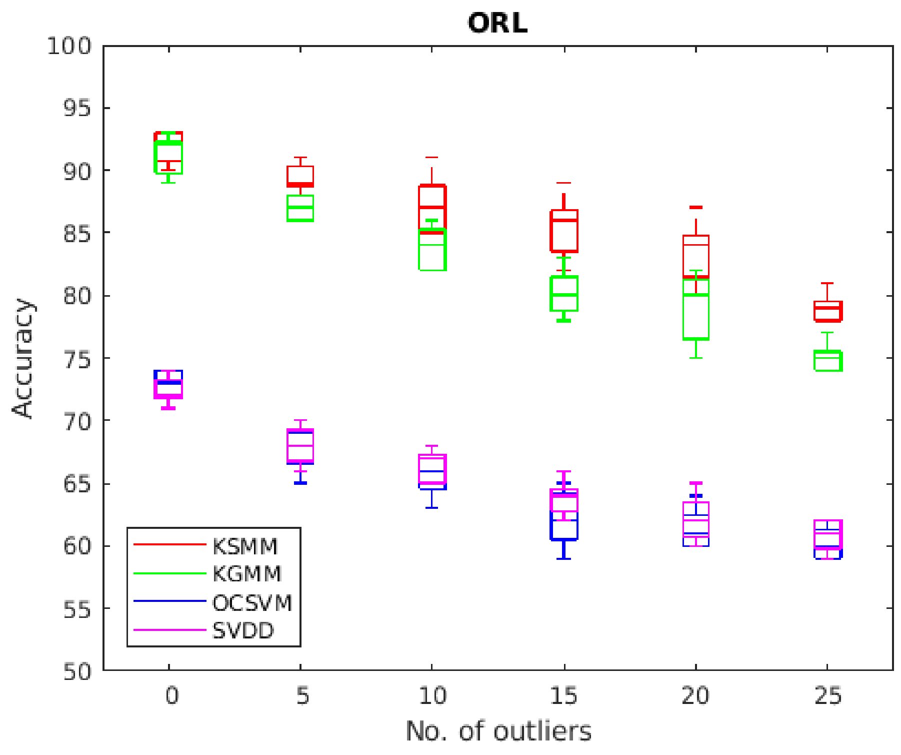


Figure 2.3: Accuracy with varying number of outliers for ORL dataset

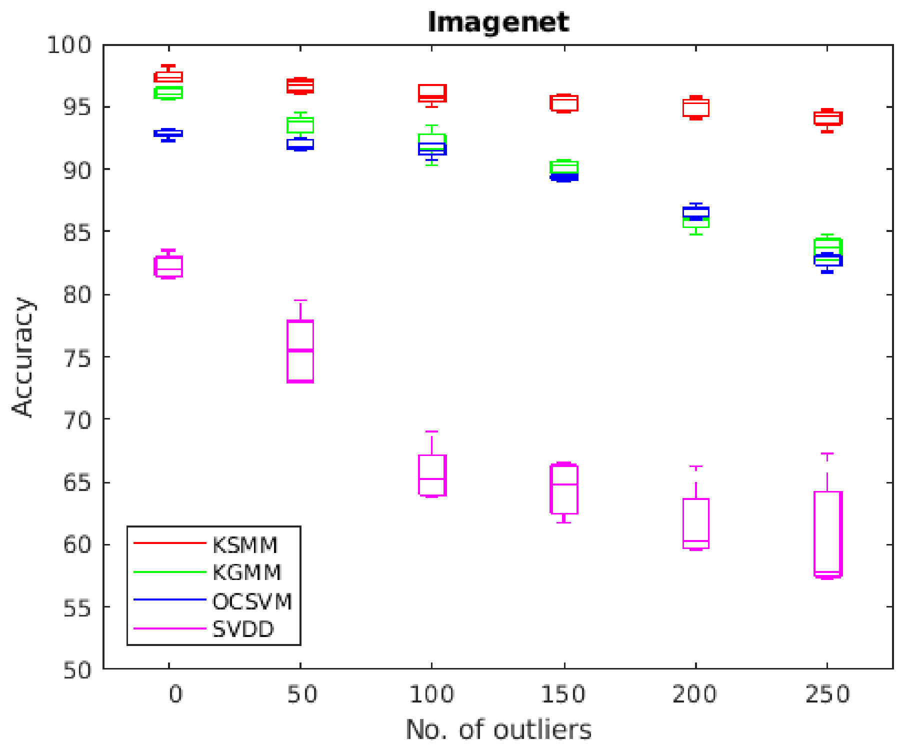


Figure 2.4: Accuracy with varying number of outliers for Imagenet dataset

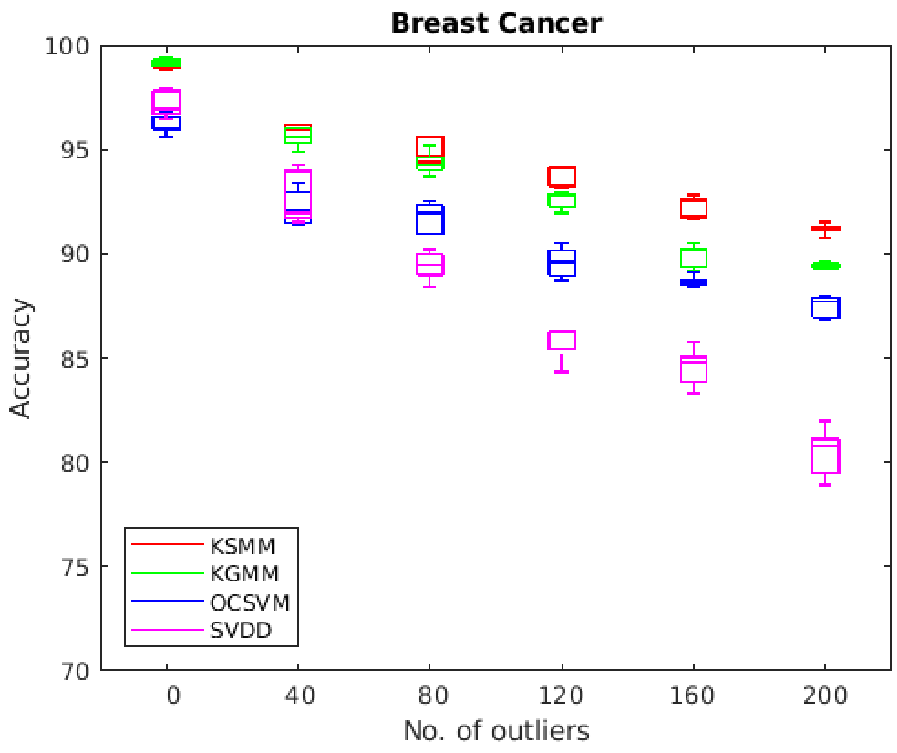


Figure 2.5: Accuracy with varying number of outliers for Breast Cancer dataset

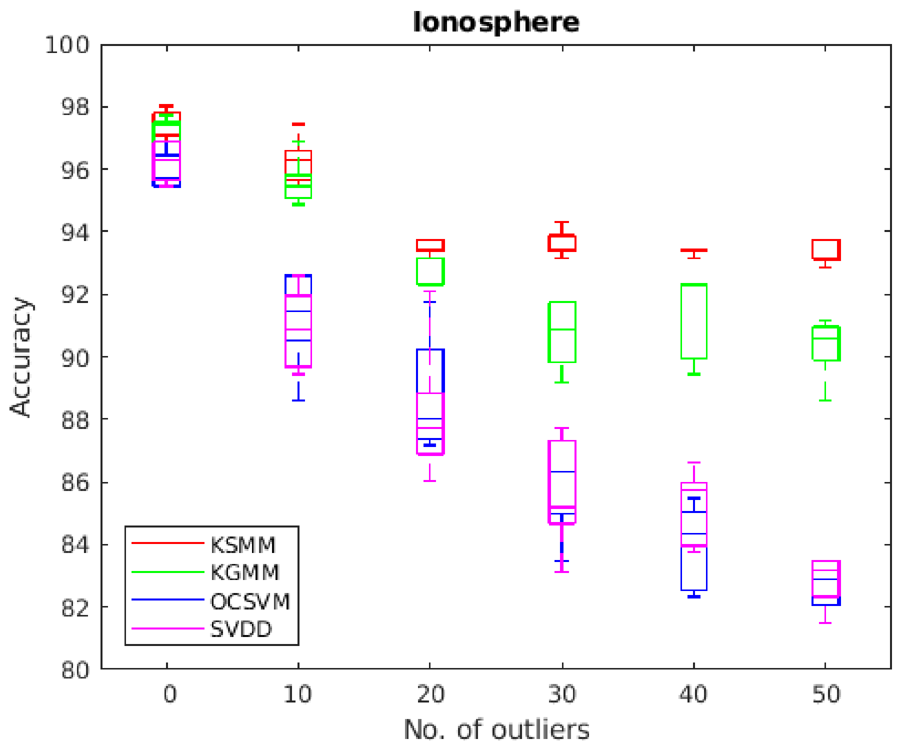


Figure 2.6: Accuracy with varying number of outliers for Ionosphere dataset

Chapter 3

Sparse Kernel Student-t Mixture Model

Sparsity is required in order to speed up the computation, especially for the cases when number of points (N) is large. Hence we propose a heuristic based approach for To incorporate sparsity in the model, we constrain eigenfunctions of Σ_l to be sparse. A naive way to do this is to keep only the largest k coefficients (by absolute value) of eigenfunction and make remaining coefficients zero after training is completed. Better way to do this is to encourage sparsity while training itself. We put a hard $L1$ norm constraint along with the sparsity constraint on eigenfunctions of Σ_l. We find the closest projection onto $L1$ ball for the obtained eigenfunctions satisfying given sparsity level. Let v be an eigenfunction whose sparse representation is to be found and w be any general vector in kernel space.

$$v = \sum_{n=1}^{N} \alpha_n \tilde{\Phi}(x_n) \tag{3.1}$$

$$w = \sum_{n=1}^{N} \beta_n \tilde{\Phi}(x_n) \tag{3.2}$$

We intend to solve the following optimization problem.

$$\beta^* = \underset{\beta}{\operatorname{argmin}} ||\alpha - \beta||_2^2 \quad \text{s.t. } ||\beta||_0 \leq \kappa \quad \text{and} \quad ||\beta||_1 = \eta \tag{3.3}$$

$$w^* = \sum_{n=1}^{N} \beta^* \tilde{\Phi}(x_n) \tag{3.4}$$

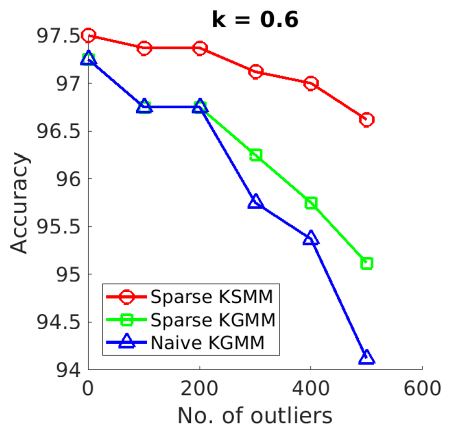
The optimization is solved using the method explained in Kyrillidis *et al.* (2013). η is tuned according to the sparsity level (κ). We summarize the algorithm for Sparse Kernel Student-t Mixture Model below 2

```
**Input**: A set of points $\{x_n\}_{n=1}^N$ in input space. Gram matrix $G$ underlying a kernel
  such that $G(i,j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$. Number of clusters $g$, degree of freedom
  parameters $\{\nu_l\}$ . Set iteration number $t = 0$.
**Initialization**;
  1. Initialize $\tau_{li}$ such that $\sum_{l=1}^{g} \tau_{li} = 1 \; \forall \; i$
  2. Initialize $w_{li}$. It is better to initialize all $w_{li}$ with 1, because it would initialize like a
     KGMM
**while** *stopping criterion == false* **do**
    1. Compute $\{\pi_l\}$, $\{a_{li}\}$ and $\{b_{li}\}$ from the updates mentioned earlier using old values of
       $\{\tau_{li}\}$ and $\{w_{li}\}$.
    2. Compute matrices $\{\tilde{\mathcal{K}}_l\}$ and their eigenvectors and eigenvalues
    3. Compute sparse projections of eigenvectors onto $L-1$ ball parameterized by $\eta$.
    4. Compute $\{\delta_{li}\}$ using the projected eigenvectors and eigenvalues of $\{\tilde{\mathcal{K}}_l\}$ to further
       calculate $Pr(\phi(x_i); \mu_l, \Sigma_l, \nu_l)$.
    5. Update $\{\tau_{li}\}$ and $\{w_{li}\}$
    6. Check stopping criterion for convergence i.e. either $t > t_{max}$ or
       $\sum_{i=1}^{n} \sum_{l=1}^{g} (\tau_{li}^{t} - \tau_{li}^{t-1})^2 < \epsilon$.
**end**
**Output**: A set of optimal parameters $\{\pi_l\}$, $\{a_{li}\}$ and $\{b_{li}\}$ representing the sparse
student-t mixture model in kernel space.
```

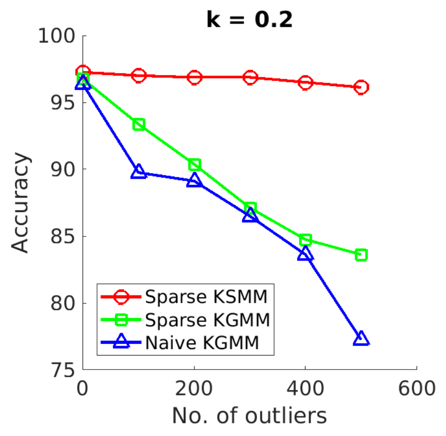
Algorithm 2: Sparse KSMM algorithm

3.1 Experimental Results

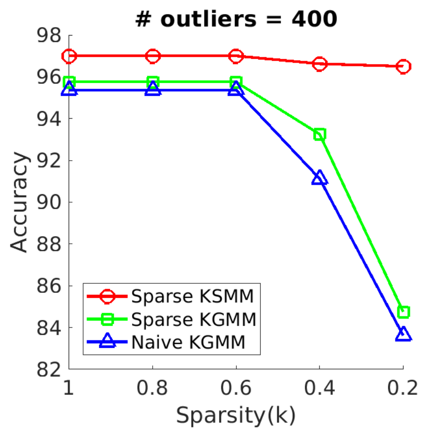
We compare our proposed Sparse-KSMM model against two other sparse models; naive KGMM and Sparse-KGMM. In naive KGMM method, we do not make any changes in the training method. After training is finished and we have obtained the optimal coefficients of eigenfunctions, we keep top κ coefficients with highest absolute value and set rest of them to 0. For Sparse-KGMM, we solve the above mentioned optimization problem in each iteration of training like Sparse-KSMM. Training and test datasets are similar to previous chapter. We show results with varying number of outliers for a given sparsity level and varying sparsity for a given number of outliers.



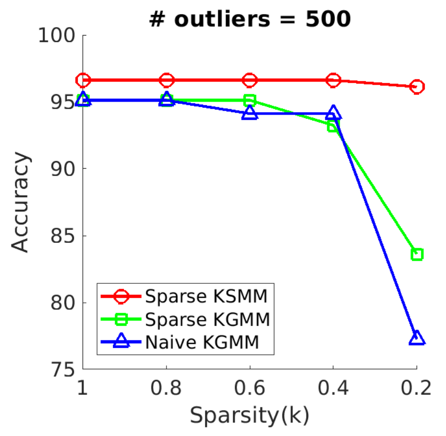
(a) Varying number of outliers for 0.6 sparsity



(b) Varying number of outliers for 0.2 sparsity

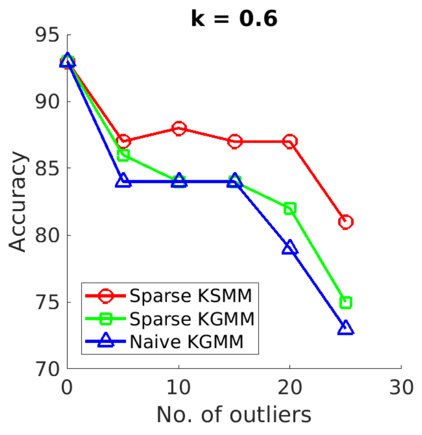


(c) Varying sparsity for 400 outliers

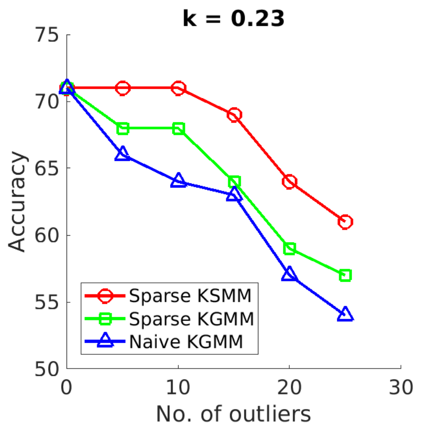


(d) Varying sparsity for 500 outliers

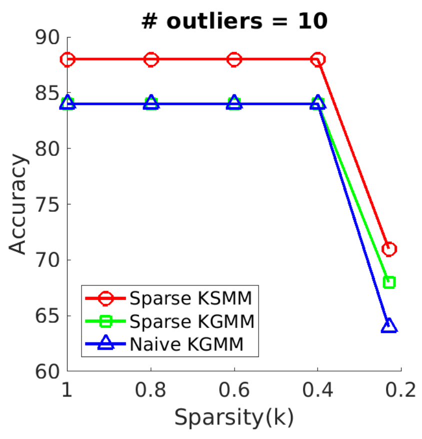
Figure 3.1: Accuracy of sparse models on MNIST dataset



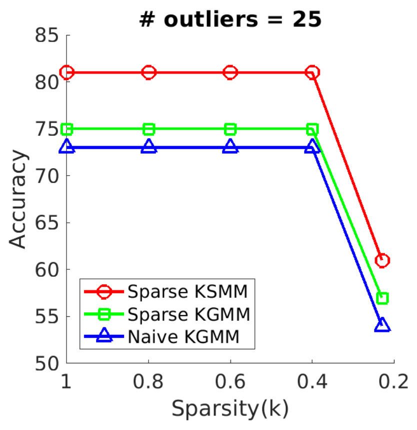
(a) Varying number of outliers for 0.6 sparsity



(b) Varying number of outliers for 0.23 sparsity

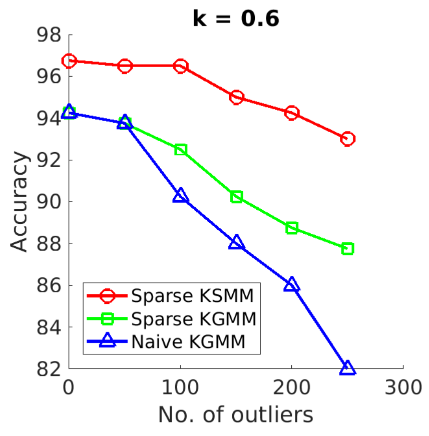


(c) Varying sparsity for 10 outliers

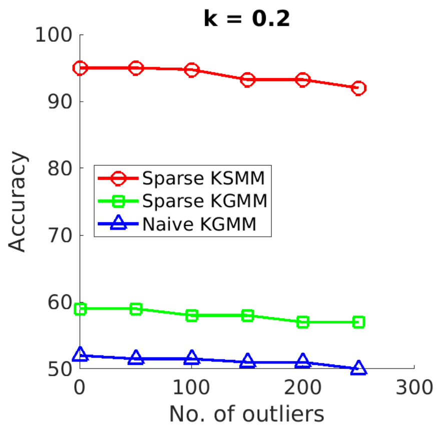


(d) Varying sparsity for 25 outliers

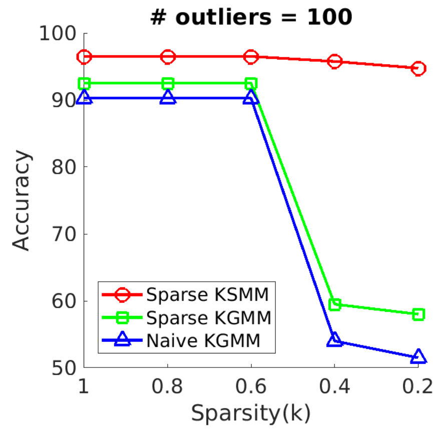
Figure 3.2: Accuracy of sparse models on ORL dataset



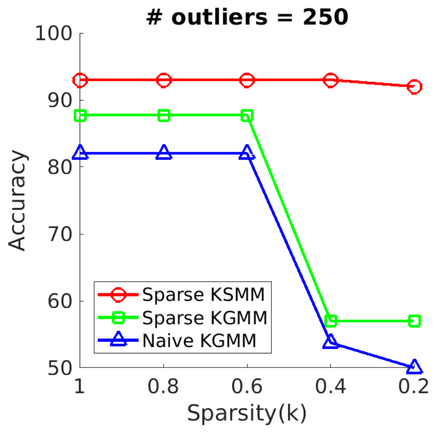
(a) Varying number of outliers for 0.6 sparsity



(b) Varying number of outliers for 0.2 sparsity

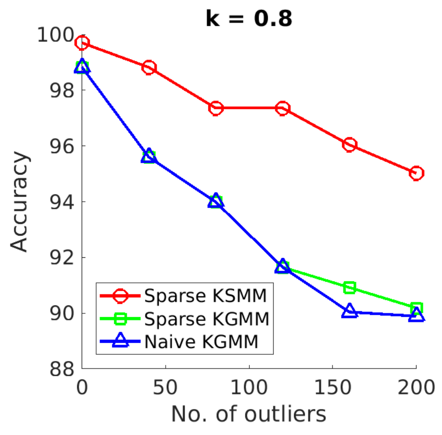


(c) Varying sparsity for 100 outliers

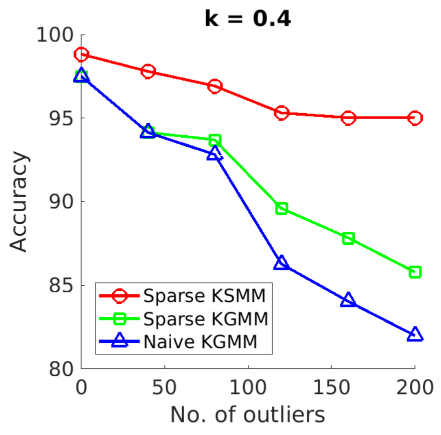


(d) Varying sparsity for 250 outliers

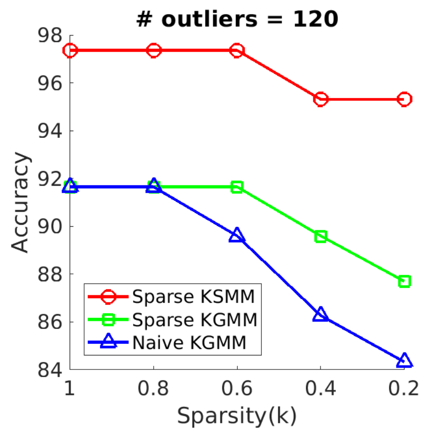
Figure 3.3: Accuracy of sparse models on Imagenet dataset



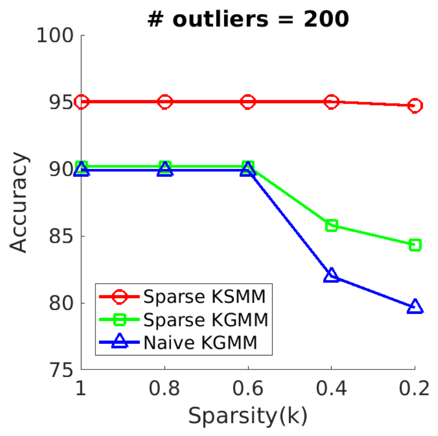
(a) Varying number of outliers for 0.8 sparsity



(b) Varying number of outliers for 0.4 sparsity

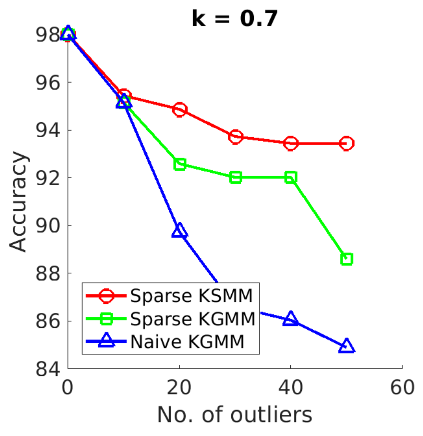


(c) Varying sparsity for 120 outliers

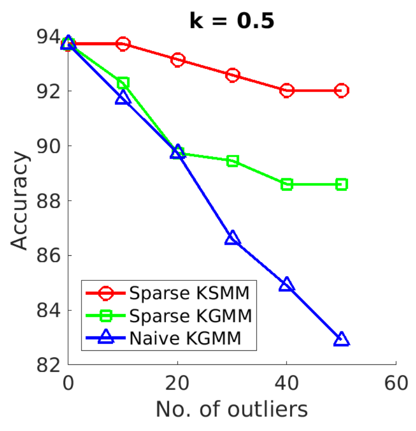


(d) Varying sparsity for 200 outliers

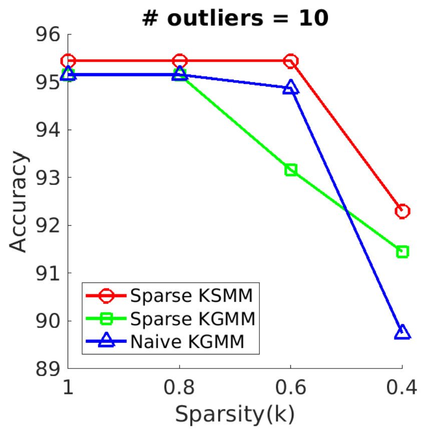
Figure 3.4: Accuracy of sparse models on Breast Cancer dataset



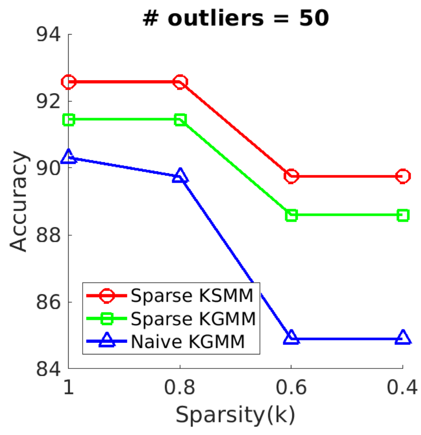
(a) Varying number of outliers for 0.7 sparsity



(b) Varying number of outliers for 0.5 sparsity



(c) Varying sparsity for 10 outliers



(d) Varying sparsity for 50 outliers

Figure 3.5: Accuracy of sparse models on Ionosphere dataset

Chapter 4

Kernel Principal Geodesic Analysis for SMM

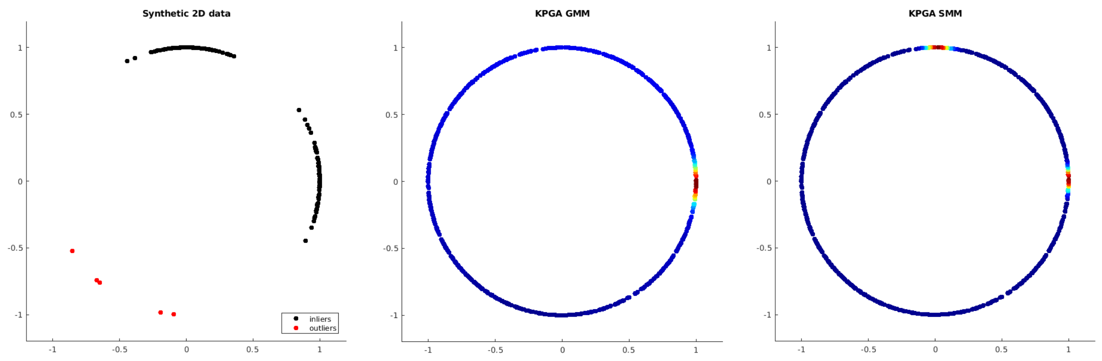
For certain kernels input points are mapped to a hyper-sphere in RKHS. This happens because for such kernels $\langle \Phi(x), \Phi(x) \rangle_{\mathcal{H}} = 1$ i.e. the dot product in kernel space of a point with itself is 1 (or a constant). Hence we have $G(i,i) = 1 \; \forall i$ for such kernels (radial basis kernel, exponential kernel etc.). Awate *et al.* (2014) have proposed method of kernel based PCA on a Hilbert sphere, called as Kernel Principal Geodesic Analysis (KPGA). The basic idea behind KPGA is to project points present on the Hilbert sphere to a hyperplane tangential to the sphere at the mean (μ) analyze sample covariance of those projected points. The projection is carried out using Logarithmic Map as follows:

$$\mathrm{Log}_{\mu}(a) = \frac{x - \langle x, a \rangle_{\mathcal{H}} \mu}{\|x - \langle x, a \rangle_{\mathcal{H}} \mu\|_2} \arccos(\langle x, a \rangle_{\mathcal{H}}) \tag{4.1}$$

Awate *et al.* (2014) has proposed a GMM model on a Hilbert sphere (KPGA-GMM). We expect that similar to SMM in Eucledean space, SMM on a Hilbert sphere (KPGA-SMM) will be more robust to the outliers than KPGA-GMM.

4.1 Experimental Results

Here we show results on a simulated dataset. We generate two clusters, each containing 100 points with normally distributed angles on a circle around respective means (0 and $\frac{\pi}{2}$) as shown in Fig. 4.1a. Then we add outliers to the data by adding points with angles far away from the means. Points corresponding to black colour are inliers while the ones belonging to red are outliers. We use linear kernel i.e. $k(x,y) = \langle x, y \rangle$. We try to fit both KPGA-GMM and KPGA-SMM on this data. The final probability distribution on the uniformly distributed points on the circle is shown in 4.1b and 4.1c respectively. It



(a) Synthetic 2D data (b) KPGA-GMM (c) KPGA-SMM

Figure 4.1: KPGA-GMM vs KPGA-SMM on synthetic 2D dataset containing outliers

can be observed that in the presence of outliers, KPGA-GMM breaks down completely showing only one peak at zero and no peak near $\frac{\pi}{2}$. But KPGA-SMM still performs well and is able to model the distribution of on the circle correctly. Hence it can be inferred that KPGA-SMM is more robust to the outliers than KPGA-GMM.

Chapter 5

Robust SMM Prior

5.1 Introduction

In this section we will briefly introduce to the fMRI and how does prior plays an important role in reconstruction of fMRI. Functional Magnetic Resonance Imaging (fMRI) is a technique used in neuroimaging to observe patterns in activations of various parts of brain. fMRI is basically a 4 dimensional object i.e. 3 dimensional voxel data collected at various time instants. Inherently the fMRI data is very noisy, hence denoising of fMRI is a crucial step. There has been multiple approaches discussed in the literature for this. Here we will focus on a Maximum Aposteriori Probability (MAP) estimation technique for fMRI reconstruction.

We need a good quality prior if we want our MAP estimate to be good. It is intuitive to choose a mixture distribution as a model for prior in fMRI data because of the various parts of the activity various clusters present in the brain. Generally a prior is supposed to be learned from a clean fMRI image and then used with denoising of other noisy images, but such clean images are not readily available. Generally such images contain outliers. Hence one needs to device a robust technique to learn prior. That is why SMM can be used here as a prior over fMRI data.

5.2 Reconstruction Scheme

5.2.1 Problem Formulation

We define a time series of a particular voxel as our data point i.e. a unit for data will be a T dimensional feature vector for each voxel, where T is the total number of frames collected for that fMRI. Let $Y = \{y_1, y_2, y_3 \cdots, y_N\}$ where y_j is the T dimensional observed time series feature vector for j^{th} voxel. Similarly, let $X = \{x_1, x_2, x_3 \cdots, x_N\}$

where x_j is the T dimensional actual time series feature vector for j^{th} voxel. Our aim is to estimate X from the observed Y. The problem can be constructed as a Maximum Aposterior Probability (MAP) estimation of X. Here we will be using two priors. First, MRF smoothness prior in order to put the smoothness constraints on X. Second is mixture distribution prior. As we had explained earlier, we assume that these time series are generated from a mixture distribution, whose parameters are learned already. We assume log prior probability ($\log Pr(X)$) as a linear combination of these two factors with weighing α and β being the weighing coefficients respectively. These α and β are required to be tuned appropriately in order to achieve optimal results. Mathematically the problem can be formulated as follows.

$$\begin{aligned}
Pr(X|Y) &= \frac{Pr(Y|X)Pr(X)}{Pr(Y)} \\
\underset{X}{max} \ \log(Pr(X|Y)) &\equiv \underset{X}{min} - \log(Pr(X|Y)) \\
&\equiv \underset{X}{min} \ - \log(Pr(Y|X) - \log(Pr(X)) \\
-\log(Pr(Y|X) &= \text{Data Fidelity loss} \\
-\log(Pr(X)) &= \alpha * \log \text{MRF misfit} + \beta * \log \text{mixture distribution misfit}
\end{aligned}$$

Traditionally GMM has been used as a mixture distribution whose parameters are learned from the training data. But this training data itself may have outliers, which will cause GMM to learn wrong parameters and thus incorrect reconstruction when we are using it for other fMRI data. If we use SMM instead of GMM, it is expected to learn parameters correctly even in the presence of large outliers and thus provide us a correct reconstruction.

5.2.2 Reconstruction Algorithm

There are two steps involved here. First step is to learn the prior and second step is to reconstruct from a corrupted image using that prior. We estimate parameters of mixture distributions (GMM or SMM) using EM algorithm from the data in the first step. After that we use gradient descent for reconstruction of the test data.

```
Gradient descent algorithm for reconstruction
Initialize X = Y
$\alpha$ (learning rate) = constant
while stopping criterion == false do
    1. Compute gradient $\frac{dlog(Pr(X|Y))}{dX}$
    2. Predict next $X_{est}$ equals to $X - \alpha * \frac{dlog(Pr(X|Y))}{dX}$
    3. Compute new objective $-log(Pr(X_{est}|Y))$
    if new objective < old objective then
        Update $X = X_{est}$
        increase $\alpha$;
    else
        decrease $\alpha$;
    end
end
```

Algorithm 3: Gradient descent algorithm for reconstruction

5.3 Experimental Results

We demonstrated results on a synthetic fMRI data and showed the SMM prior causes a better reconstruction than traditional GMM prior. First we will describe the experimental setup and then results obtained using both the algorithms.

5.3.1 Experimental Setup

1. We simulated fMRI data using a 2D image whose intensity was varied in time. We used Shepp Logan phantom image having 3 clusters. The intensity of each cluster was varied sinusoidally. Each cluster was assigned a different phase, the frequency of sinusoid was chosen that 10 time frames would correspond to one full time period. Thus we obtained a 10 dimensional time series feature vector for each pixel in the image.

2. We added Gaussian noise with diagonal covariance matrix to simulate real life corruption of the image due to data collection process.

3. Then we added salt and pepper noise to simulate outliers. Salt and pepper noise fixes values of randomly chosen pixels to zero or one, which causes significant de-

viation from the usual sinusoidally smoothly varying data. This very well simulates the outliers.

4. After that we learn parameters of GMM and SMM with 3 components from this corrupted data containing outliers. These learned models will be used later in the reconstruction.

While reconstruction following losses and data corruption models were used. The optimization was done using gradient descent algorithm.

1. Data Fidelity Loss
   We use simple Gaussian noise model as for data corruption. Hence -log $Pr(y_j|x_j)$ simply becomes proportional to $|y_j - x_j|^2$.

2. Log MRF misfit
   MRF prior is very common in putting smoothness constraints on data. We used 4 neighborhood squared error model for MRF. Therefore log MRF misfit becomes $\sum_{p\in\{-1,1\}}\sum_{q\in\{-1,1\}} |x_{i,j} - x_{i+p,j+q}|^2$. Note that subscript notation here has been different than before, just for the notational simplicity.

3. Log Mixture Distribution misfit This is nothing but negative log probability under the given mixture model. It simply becomes $-\log \sum_{l=1}^{3} \alpha_l Pr(x_j|\theta_l)$ for j^{th} datapoint. $Pr(x_j|\theta_l)$ will be given by Gaussian distribution or Student-t distribution in case of GMM and SMM respectively. θ_l are parameters of given distribution, which are learned from prior.

5.3.2 Prior Parameters Results

- Estimated Means
  Figure 5.1 shows true means, means estimated by GMM and SMM for various clusters. We found out that in presence of outliers means learned by GMM are wrong, especially for the cluster denoted by yellow color. We also calculated Euclidean distance between means estimated by GMM and SMM with true means. As expected means estimated by SMM were closer to the true means. The results are mentioned in Table 5.1.

- Estimated Covariances
  Similar to the case of estimated means, covariance matrices are also estimated correctly by SMM than GMM. We used two measures for quantifying the difference;

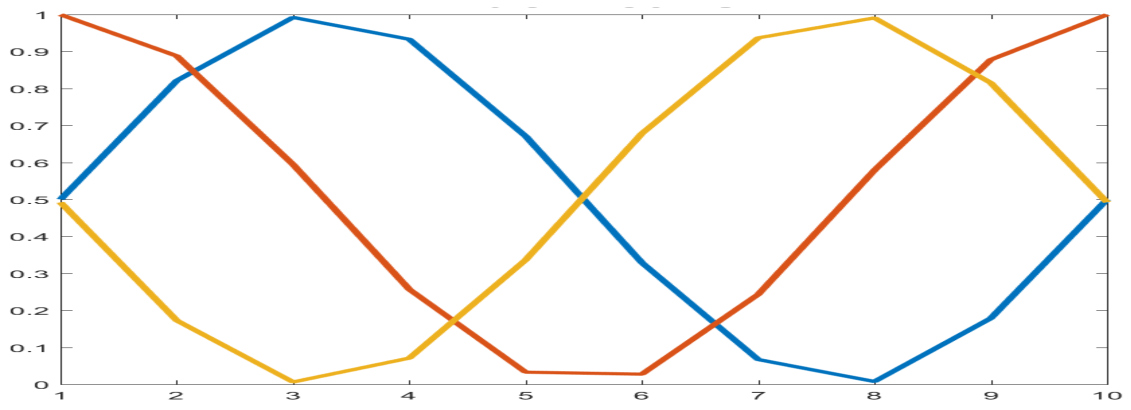
average Frobenius distance from the original covariance matrices and average absolute difference between the eigenvalues. SMM has performed better than GMM in terms of both the criteria. The results are mentioned in Table 5.1. $\lambda_{t,l}$ represents t^{th} eigenvalue of l^{th} cluster.

Quantity	Value	SMM	GMM
Avg. Euclidean distance between cluster means	$\frac{1}{3}\sum_{l=1}^{3} \|\|\mu_l^{est} - \mu_l^{orig}\|\|_2$	0.0074	0.3651
Avg. Frobenius distance between cluster covariance matrices	$\frac{1}{3}\sum_{l=1}^{3} \|\|\Sigma_l^{est} - \Sigma_l^{orig}\|\|_{frobenius}$	0.0209	0.3029
Avg. absolute difference between eigenvalues of covariance matrices	$\frac{1}{3}\sum_{l=1}^{3}\sum_{t=1}^{10} \|\lambda_{t,l}^{est} - \lambda_{t,l}^{orig}\|$	0.0631	0.6468

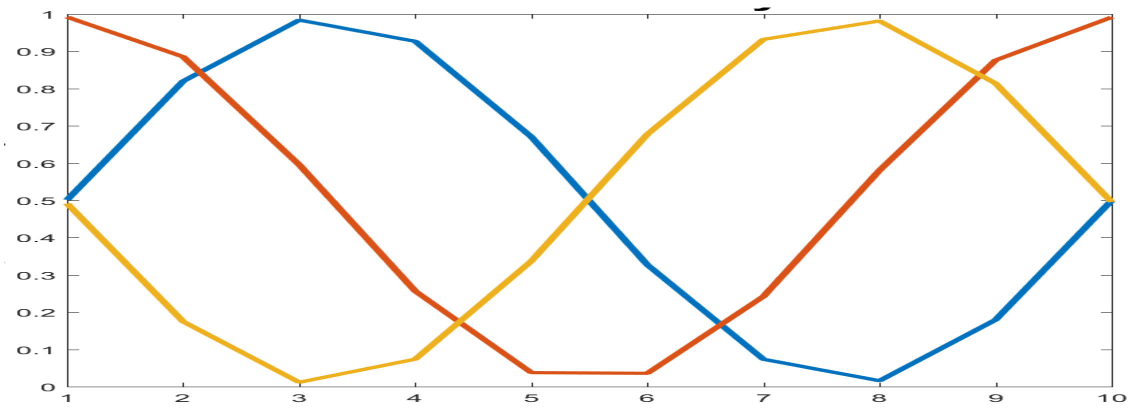
Table 5.1: GMM vs SMM quantitative results for parameter estimation

5.3.3 Reconstruction Results

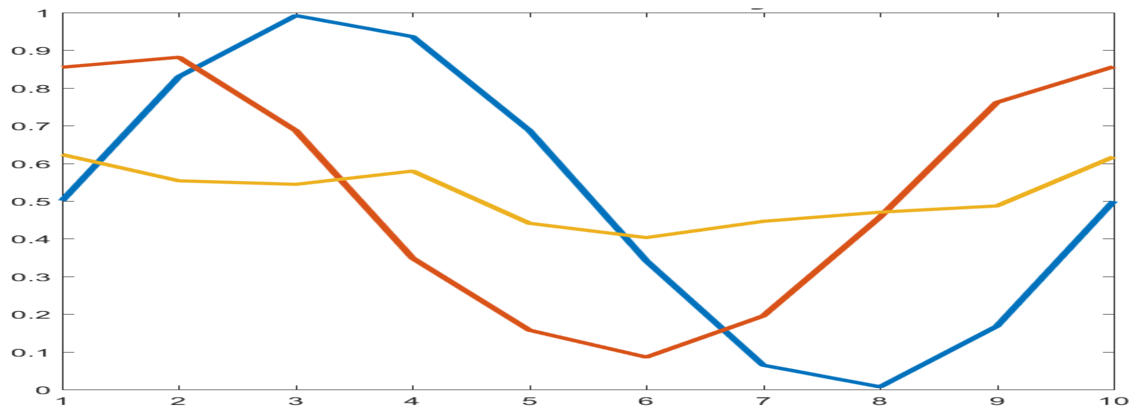
Both the priors (GMM and SMM) were used to reconstruct a Gaussian noise corrupted image also having salt and pepper outliers. Frame #3 of original 5.2a, corrupted 5.2d, reconstructed using SMM MRF prior 5.2b, reconstructed using GMM MRF prior 5.2e, absolute difference with the original image 5.2c,5.2f are shown in Figure 5.2. Similar results for frame#8 are shown in Figure 5.3 It can easily be concluded that reconstruction using SMM-MRF prior is really close to the original image, while the one using GMM-MRF prior is far away from the original image. Note that hyperparameters were tuned to obtain the optimal results in both the cases. Entire video of 10 frames is available here.



(a) True Cluster Means



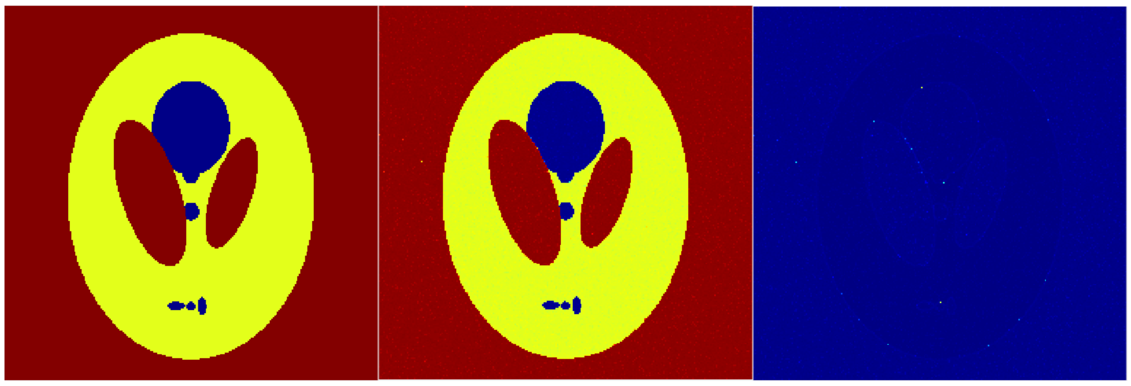
(b) SMM Cluster Means



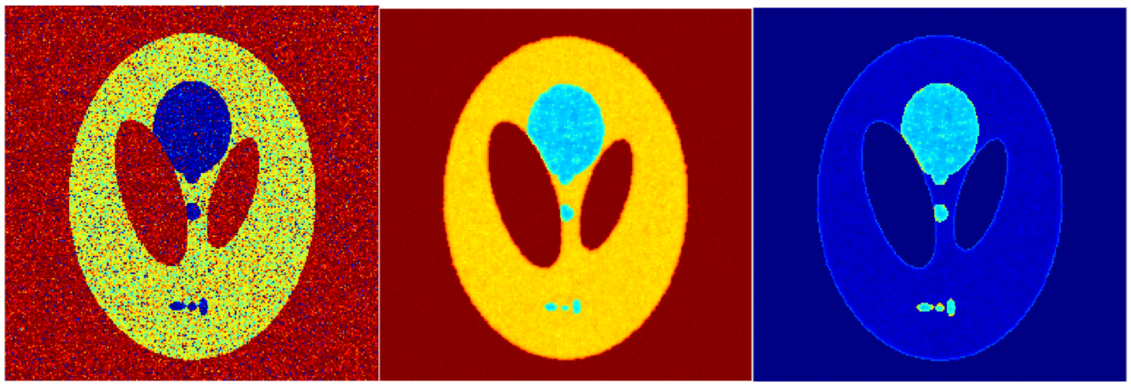
(c) GMM Cluster Means

time (frame #)

Figure 5.1: Means learned by GMM and SMM from corrupted data containing outliers

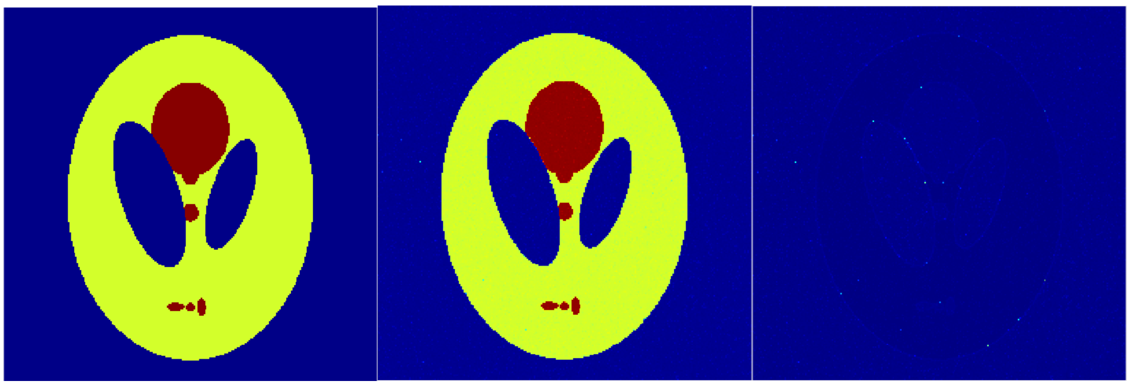


(a) Original data (b) SMM-MRF reconstructed (c) SMM-MRF absolute diff.

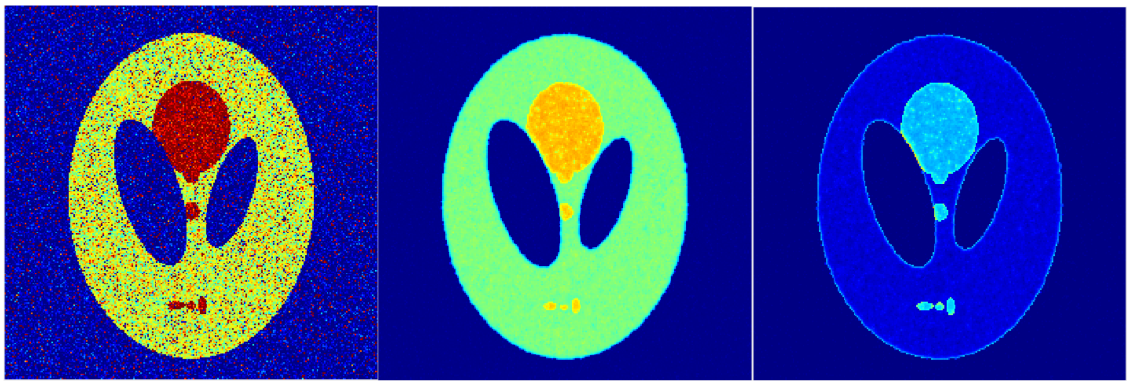


(d) Corrupted data (e) GMM-MRF reconstructed (f) GMM-MRF absolute diff.

Figure 5.2: Reconstruction results for frame #3 SMM vs GMM



(a) Original data (b) SMM-MRF reconstructed (c) SMM-MRF absolute diff.



(d) Corrupted data (e) GMM-MRF reconstructed (f) GMM-MRF absolute diff.

Figure 5.3: Reconstruction results for frame #8 SMM vs GMM

Chapter 6

Conclusion and Future Work

We have shown that robustness can be incorporated in kernel spaces using our formulation of KSMM. The results demonstrated on clustering and outlier detection has shown that it clearly outperforms baseline KGMM and other one class classification methods of outlier detection. Current EM framework for KSMM developed by us assumes constant degree of freedom which needs to be tuned. It would be better if updates for degree of freedom parameter of SMM can be found using kernel tricks. Also, we would like to show results of KPGA-SMM on real datasets in future.

We have also showed that SMM work as a better prior than GMM for the reconstruction of the synthetic fMRI data. Data Fidelity loss model needs to be changed while applying on real fMRI data. Instead of calculating data fidelity in Euclidean domain, it would be beneficial to calculate it in wavelet coefficients domain. Also, instead of simple 4 neighborhood squared error MRF prior, an edge preserving smoothness prior like Huber function needs to be used.

References

Aronszajn, N., 1950, “Theory of reproducing kernels,” *Trans. Amer. Math. Soc.* **68**, 337–404.

Awate, S. P., Yu, Y.-Y., and Whitaker, R. T., 2014, “Kernel principal geodesic analysis,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (Springer). pp. 82–98.

Bishop, C., 2007, “Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn,” *Springer, New York*

Boser, B. E., Guyon, I. M., and Vapnik, V. N., 1992, “A training algorithm for optimal margin classifiers,” in *Proceedings of the fifth annual workshop on Computational learning theory* (ACM). pp. 144–152.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., 2009, “Imagenet: A large-scale hierarchical image database,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (IEEE). pp. 248–255.

Hennig, C., 2004, “Breakdown points for maximum likelihood estimators of location-scale mixtures,” *Annals of Statistics*, 1313–1340.

Kyrillidis, A., Becker, S., Cevher, V., and Koch, C., 2013, “Sparse projections onto the simplex,” in *International Conference on Machine Learning*, pp. 235–243.

LeCun, Y., Cortes, C., and Burges, C., 2010, “Mnist handwritten digit database,” *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* **2**

Manevitz, L. M., and Yousef, M., 2001, “One-class svms for document classification,” *Journal of machine Learning research* **2**, 139–154.

McLachlan, G. J., Ng, S.-K., and Bean, R., 2016, “Robust cluster analysis via mixture models,” *Austrian Journal of Statistics* **35**, 157–174.

Rayana, S., 2016, “ODDS library,”

Samaria, F. S., and Harter, A. C., 1994, “Parameterisation of a stochastic model for human face identification,” in *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on* (IEEE). pp. 138–142.

Scholkopf, B., and Smola, A., 2002, *Learning with Kernels* (MIT Press).

Schölkopf, B., Smola, A., and Müller, K.-R., 1997, “Kernel principal component analysis,” in *International Conference on Artificial Neural Networks* (Springer). pp. 583–588.

Sigillito, V. G., Wing, S. P., Hutton, L. V., and Baker, K. B., 1989, “Classification of radar returns from the ionosphere using neural networks,” *Johns Hopkins APL Technical Digest* **10**, 262–266.

Simonyan, K., and Zisserman, A., 2014, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*

Tax, D. M., and Duin, R. P., 2004, “Support vector data description,” *Machine learning* **54**, 45–66.

Wang, J., Lee, J., and Zhang, C., 2003, “Kernel trick embedded gaussian mixture model,” in *ALT* (Springer). pp. 159–174.

Wolberg, W. H., and Mangasarian, O. L., 1990, “Multisurface method of pattern separation for medical diagnosis applied to breast cytology..” *Proceedings of the national academy of sciences* **87**, 9193–9196.

Acknowledgements

I would like to thank Prof. Suyash Awate and Prof. S. N. Merchant for providing me this opportunity and express my gratitude to them for the constant support.

Kalpesh Patil
IIT Bombay
9 June 2018