Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı, S. 26-37, Ağustos 2020
© Telif hakkı EJOSAT'a aittir
Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue, pp. 26-37, August 2020
Copyright © 2020 EJOSAT
Research Article

Sınıflandırıcı Performanslarının Gauss Karışım Modeline Uygulanan Beklenti-Maksimizasyon Algoritmasına Göre Analiz Edilmesi

Korhan Cengiz[1]

[1] Trakya Üniversitesi, Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, Edirne, Türkiye (ORCID: 0000-0001-6594-8861)

(Bu yayın 26-27 Haziran 2020 tarihinde HORA-2020 kongresinde sözlü olarak sunulmuştur.)

(**DOI:** 10.31590/ejosat.)

ATIF/REFERENCE: Cengiz, K. (2020). Sınıflandırıcı Performanslarının Gauss Karışım Modeline Uygulanan Beklenti-Maksimizasyon Algoritmasına Göre Analiz Edilmesi. *Avrupa Bilim ve Teknoloji Dergisi*, (Special Issue), 26-37.

Öz

Maksimum olabilirlik, karışım modeli, bayes sonucu ve maksimum entropi gibi parametric yoğunluk kestirimleri dağılımın çeşidi bilindiğinde veya tahmin edilebilir olduğunda sıklıkla kullanılmaktadır. Beklenti maksimizasyonu veya değişken adım öğrenme algoritması dağılım parametrelerinin maksimum olabilirliğini elde etmenin en başarılı yollarıdır. Bu makalede, üç farklı dağılım içeren çok boyutlu Gauss karışım modeline EM algoritmasının uygulanması amaçlanmıştır. Bu çalışmada istatistiksel dağılım, Gauss dağılımından elde edilmiştir ve her dağılım için ortalama ve kovaryans matrisi olan parametreler tahmin süreci için kullanılmıştır. Orijinal özellik vektörleri ve onların tahminleri benzerlik açısından karşılaştırılmış aynı zamanda elde edilen sonuçlar sunulmuş ve detaylı bir şekilde tartışılmıştır. Ek olarak, çatallı veri kümesi için her bir dağılım belirtilmiştir. Son olarak, Bayes, k-NN ve diskriminant sınıflandırma metotları GMM' ye uygulanmış ve bu metotların performansları analiz edilmiştir.

Anahtar Kelimeler: Bayes Sınflandırması, Yoğunluk Tahmini, EM Algoritması, GMM, k-NN ve LDA.

Analyzing Classifier Performances Based on Implemented Expectation-Maximization Algorithm to Gaussian Mixture Model

Abstract

Parametric density estimations i.e., maximum likelihood, mixture model, bayesian inference, maximum entropy are frequently used when type of distribution is known or predictable. Expectation-Maximization (EM) or a variable step learning algorithm are most successful ways for obtaining maximum likelihoods of distribution parameters. In this paper, we aim to present implementation of the EM algorithm to multidimensional Gaussian mixture model (GMM) that includes three different distributions. In this study, the statistical distribution is obtained from Gaussian distribution and parameters which are mean and covariance matrices for each distributions are used for estimation process. Original feature vectors and their estimates are compared in term of similarity as well as obtained results are presented and discussed in details. In addition, each distribution for bifurcated dataset is indicated. Finally, Bayesian, k-NN and Discriminant classifier methods are implemented to GMM and the performance of these methods are analyzed.

Keywords: Bayesian Classfication, Density estimation, EM Algorithm, GMM, k-NN, LDA.

[1] Sorumlu Yazar: Trakya Üniversitesi, Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, Edirne, Türkiye, ORCID: 0000-0001-6594-8861, korhancengiz@trakya.edu.tr

1. Introduction

Density estimation is commonly used in statistical theory. It forms the significant part of classification methods and pattern recognition problems. In general, density estimation techniques can be classified into two major groups that are parametric and non-parametric estimation [1]. In non-parametric methods such as parzen windows and k-NN, there is no information about the type of the distribution and its parameters. In this case, probability density function should be found on related point and then classification should be made according to the classification conditions which vary with the type of the classification design. However, in parametric methods which are maximum likelihood, bayesian inference, maximum entropy and mixture model, it is assumed that type of the distribution is known and computed related parameters vary according to the type of the distribution. When recent studies [2, 3] about mixture model are examined, it can be attained that, used datasets are not observable in 2D and 3D space [4].

Consequently, this makes difficult to preprocess states such that non-pre-prediction of data or ignoring outlier process before implementing EM algorithm. Thus, this study aims to estimate unknown parameters of data which are assumed to be composed of more than one distributions, and also it can be denoted by two features. In this work, we aim to present application of the EM algorithm in order to estimate distribution parameters which have mixture model and the accuracy of found parameters are computed. During this process, datasets are supposed to be as mixture model and all distributions including training dataset are Gaussian. Also, computed means and covariances of each distribution are obtained via a varying step learning algorithm and predicted parameters. In evaluations, estimated parameters and their consisted distributions are compared with original distribution. Also, some classifiers are implemented to two different datasets. In order to evaluate the accuracy of the classifier, classifiers are implemented to GMMs and they are compared by the help of estimated parameters and prior probabilities. Moreover, the contribution amount of training data and test data to the classifier performance are determined.

2. Mixture Model

Mixture model (MM) is the fundamental part of parametric density estimation schemes that based on assuming any type of distribution which can be Gaussian, Exponential, Rayleigh etc., or combinations of them. By a majority, this model is used when information about the type of the distribution does not exist or data are association of multiple distributions [5]. MM is commonly used in signal and video processing applications. This model especially uses when data is observable or it can be expressible in 2D space. The main goal of this method is density function estimation which is commonly used for classifier design or pattern recognition. In general, any probability density distribution (pdf) which has two parameters can be expressed by (1):

$$p_k(\mathbf{x}\mid\varphi)=\sum_{k=1}^{K}\omega_{i,k}\,p_k(\mathbf{x}\mid Z_k,\theta_k) \qquad (1)$$

where K is number of elements of distribution in dataset, φ includes two parameters and Z_k and θ_k denote to symbolic parameters. In (1), w_{ik} corresponds to membership weight/probability and it can be expressed as:

$$\omega_{i,k}=\frac{p_k(\mathbf{x}_i|Z_k,\theta_k)}{\sum_{m=1}^{K}p_m(\mathbf{x}_i|Z_m,\theta_m)\alpha_m} \qquad (2)$$

where N represents the number of feature vectors in mixture model and α_m denotes the probability of each distribution. The number of N does not have significant impact on parameter estimation process. However, during designing of classification, it becomes very precious, because if so many data is used in order to train the classifier, accuracy of classifier will increase automatically.

$$\sum_{k=1}^{K}\omega_{i,k}=1 \qquad (3)$$

On the other hand, increase in the number of N will increase computational process of the algorithm. Membership weight vector specifies correlation of each vector between another feature vectors under Bayesian rule. It should be noted that, since membership weight vector represents all probability of feature vectors.

3. Expectation-Maximation Algorithm

The EM algorithm is any type of parametric density estimations which is an iterative method for the purpose of finding maximum likelihood (ML) or maximum a-posteriori (MAP) of parameters. The algorithm consists of four steps as follows:

- Initialize
- E-Step
- M-Step
- Go to E-step

Initialize step can be started with two ways which are determination of initial parameters and usage of prior probabilities to compute randomly E-step for each class or determine initial parameters [6-8]. It should be noted that if there is any chance of pre-reviewing of the data, parameters can be selected with sense of proportion instead of random determination. Thus, if the selected values initially divergence from actual value, process load and iteration numbers of the algorithm will increase.

In E-Step, assumed Z_k and θ_k which correspond to mean and covariance matrices respectively in this study are given or they are randomly selected and computed from membership weight vector in (3). In M-Step, it is focused on finding parameters. In order to estimate prior probability of each classes, from (3), it is derived at each iteration as follow:

$$N_k = \sum_{i=1}^{N} \omega_{i,k} \tag{4}$$

In here, $k=1,2,\ldots,K$ and $i=1,2,\ldots,N$ where K shows the number of classes. In (4), N_k denotes number of feature vectors for each classes. As a result, the updated prior probabilities of classes can expressed below:

$$\alpha_k^{new} = \frac{N_k}{N} \tag{5}$$

After above derivations, maximum likelihood of Gaussian parameters can be found. Here, $\boldsymbol{x}=\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots\ldots, \boldsymbol{x}_N\}$ is assumed as Gaussian distribution and also joint pdf of multiple observed data can be written as:

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N \mid \varphi) = \prod_{k=1}^{N} p(\mathbf{x}_k \mid \varphi) \tag{6}$$

and its multivariate Gaussian pdf becomes,

$$\prod_{k=1}^{N} p(\mathbf{x}_k \mid \varphi) = \prod_{k=1}^{N} \left(\frac{1}{(2\pi)^{N/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x}_k - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}_k - \boldsymbol{\mu}) \right) \right) \tag{7}$$

In (6) and (7), N denotes the size of covariance matrix. In order to compute maximum likelihood of multiple observation of mean vectors, the following equations can be used.

$$\boldsymbol{\mu}_{ML} = \arg \max_{\boldsymbol{\mu}} \ln \left(\prod_{k=1}^{N} p(\mathbf{x}_k \mid \varphi) \right) \tag{8}$$

$$\ln \left(\prod_{k=1}^{N} p(\mathbf{x}_k \mid \varphi) \right) = \sum_{k=1}^{N} \left(-\frac{N}{2} \ln(2\pi) - \frac{1}{2} \ln|\Sigma| - \frac{1}{2} (\mathbf{X} - \boldsymbol{\mu})^T + \Sigma^{-1} + (\mathbf{X} - \boldsymbol{\mu}) \right) \tag{9}$$

Taking derivative of (8) with respect to $\boldsymbol{\mu}$, considering membership weight vector and setting it to zero, we obtain the following equation:

$$\frac{\partial}{\partial \boldsymbol{\mu}} \ln \left(\prod_{k=1}^{N} p(\mathbf{x}_k \mid \varphi) \right) = \boldsymbol{\mu}_k^{new} = \frac{1}{N_k} \sum_{k=1}^{K} \omega_{i,k} \mathbf{X}_k \tag{10}$$

To compute maximum likelihood of multiple observations of covariance matrix:

$$\Sigma_{ML} = \arg \max_{\Sigma} \log\left(\prod_{k=1}^{N} p(\mathbf{x}_k \mid \varphi) \right) \tag{11}$$

Rewriting the log-likelihood by using trace trick:

$$\begin{aligned} \log\left(\prod_{k=1}^{N} p(\mathbf{x}_k \mid \varphi) \right) &= -\frac{N}{2}\log|\Sigma| - \frac{1}{2}\sum_{k=1}^{N} \mathrm{Trace}\left(\Sigma^{-1} + (\mathbf{x}_k - \boldsymbol{\mu})(\mathbf{x}_k - \boldsymbol{\mu})^T\right) \\ &= -\frac{N}{2}\log|\Sigma| - \frac{1}{2}\mathrm{Trace}\left(\Sigma^{-1}\sum_{k=1}^{N}\left[(\mathbf{x}_k - \boldsymbol{\mu})(\mathbf{x}_k - \boldsymbol{\mu})^T\right]\right) \end{aligned} \tag{12}$$

Taking derivative of (10) with respect to $\sum^{-1}$ by taking into account membership weight and setting it to zero, we get:

$$\frac{\partial}{\partial \Sigma^{-1}} \log\left(\prod_{k=1}^{N} p(\mathbf{x}_k \mid \varphi) \right) = \Sigma_k^{new} = \frac{1}{N_k}\sum_{i=1}^{N} \omega_{i,k}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T(\mathbf{x}_i - \boldsymbol{\mu}_k) \tag{13}$$

At the end of M-Step, algorithm goes to E-Step and recalculate parameters for each iteration. In the algorithm, prior probabilities and parameters converge to their actual values for each iteration.

To terminate the algorithm, actual values should be very close to the estimated parameters. To check this, *l(φ)* and its logarithm are defined in equations (14) and (15) respectively;

$$l(\varphi) = \sum_{k=1}^{K} \omega_{i,k}\, p_k(\mathbf{x}_i \mid Z_k, \theta_k) \tag{14}$$

$$\log(l(\varphi)) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \omega_{i,k}\, p_k(\mathbf{x}_i \mid Z_k, \theta_k) \right) \tag{15}$$

Note that, logarithm function has monotonic increase. Thus, there is no difference between logarithmic function and original function. Here, the main purposes are observing variation in (14) and deciding when the algorithm is terminated. As a result, (15) becomes as following equation:

$$\log(l(\varphi)) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \alpha_k\, p_k(\mathbf{x}_i \mid Z_k, \theta_k) \right) \tag{16}$$

Finally, in order to terminate the algorithm the stopping criteria should be defined. To define stopping criteria, assuming that there is no significant change in *log(l(φ)).* If changes are minor and obtained results are similar, iteration stops and algorithm terminates. Therefore, parameters are estimated by the help of E-Step, M-Step and their iterations.

Note that, change value that denotes the change of (15) should be selected sensitively. If it is selected very big, the algorithm performance reduces and as a result, the accuracy of estimated parameters become farther away from its actual value. If it is determined as very low the algorithm becomes more complex, whereas it takes time to compute this change as well as iteration steps increase. Thus, it is recommended that this value should be chosen according to the importance of practice.

In this study, actual feature vectors are illustrated in Fig. 1. From this figure, it is apparent that mixture data are composed of three different Gaussian distributions that have different mean and covariance parameters.

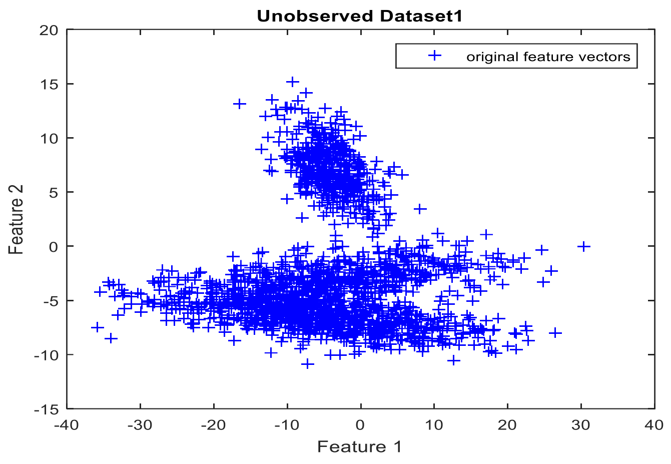

Figure 1. Unobserved Gaussian mixture model dataset which assumed that its composed of three distribution with two features

Before the algorithm is implemented, assuming that, prior probabilities of each classes are equal to each other which are *0.33*. Other parameters are defined randomly. In each iteration, (15) is recomputed until a significant change occurs in *log(l(φ))* via estimated maximum likelihood of parameters. At the end of this operation, equations (11) and (12) are slightly closer to their actual values. This process is repeated for each distribution. Then, new mixture data which are generated randomly through estimated means and covariances are redrawn and compared with raw Gaussian mixture data. The results are illustrated in Fig. 2.

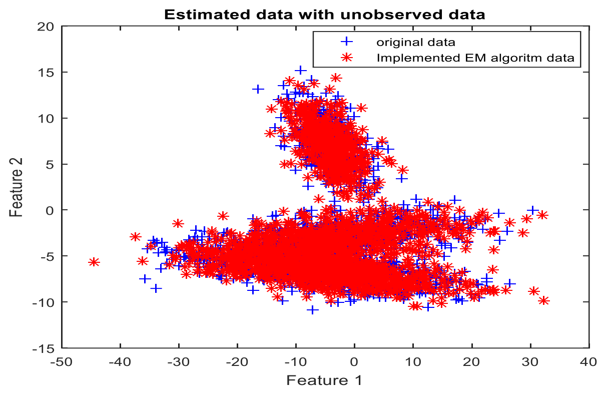

Figure 2. Original feature vectors (from dataset1) and implementation of EM algorithm to them in 2D space

Each Gaussian distribution mixture model are represented separately using estimated parameters and prior probabilities in Fig 3. This figure shows that predicted parameters are very close to their actual values. Note that, generally, in parameter estimation algorithms, when the number of feature vectors increase, the performance of the algorithm increases.

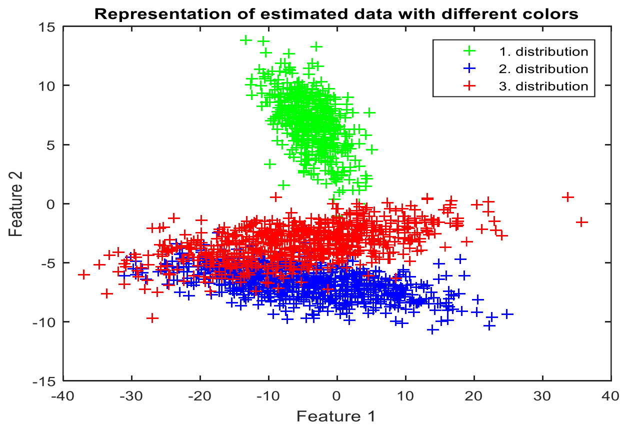

Figure 3. Representation of estimated two-dimensional Gaussian distributions

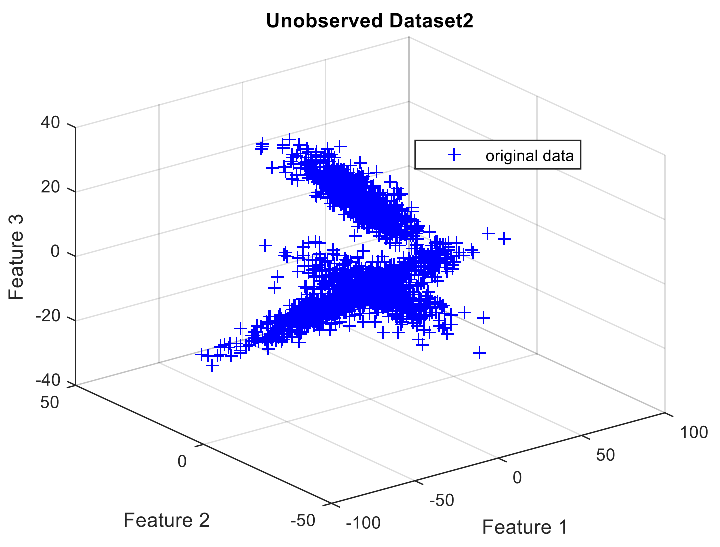

Figure 4. Unobserved Gaussian mixture model dataset which assumed that its composed of three distribution with three features

When the same process is implemented to 3D different dataset as shown in Fig. 4, estimated mean and covariance matrices are obtained. These results are illustrated in figures 5 and 6.

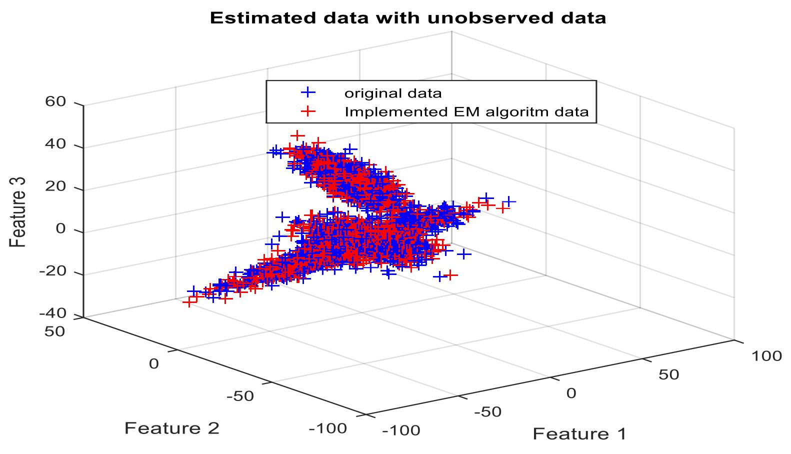

Figure 5. Original feature vectors (from dataset2) and their in case of implemented EM algorithm in three-dimensional space

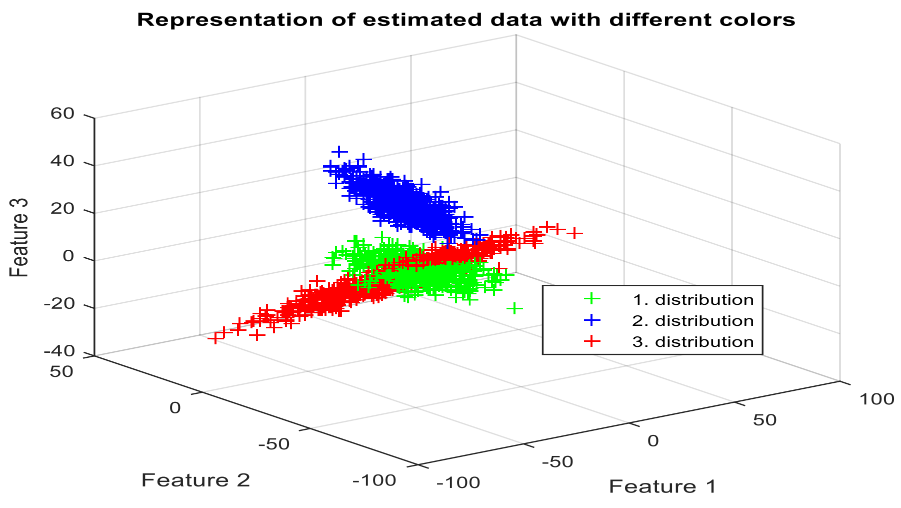

Figure 6. Representation of estimated three-dimensional Gaussian distributions

4. Class Separability Measures

Before classification, the separability of the classes should be explored. This analyze is called as Class Separability Measure (CSM). This process checks the separability of each class between other classes and gives a scalar value. If this value is a great, it means, concerning classes are separable with good accuracy. As pointed out in Bayes rule, the classification error probability depends on the log-ratio as follow:

$$D_{ij} = \ln \frac{p(\mathbf{x} \mid \omega_i)}{p(\mathbf{x} \mid \omega_j)} \tag{17}$$

(17) also gives a useful information concerning about the discriminatory capabilities associated with an adopted feature vector ***x*** and this can be used as a measure of the underlying discriminating information of class w_1 with respect to w_2. If (17) is written by taking into account the pdfs, we get:

$$D_{ij}(x) = \int_{-\infty}^{+\infty} p(\mathbf{x} \mid \omega_i) \ln \frac{p(\mathbf{x} \mid \omega_i)}{p(\mathbf{x} \mid \omega_j)} dx \tag{18}$$

For multiclass problems, the divergence is computed for every class pair ω_i and ω_j. The sum of divergences are calculated as in (19):

$$d_{ij} = \int_{-\infty}^{+\infty} (p(\mathbf{x} \mid \omega_i) - p(\mathbf{x} \mid \omega_j)) \ln \frac{p(\mathbf{x} \mid \omega_i)}{p(\mathbf{x} \mid \omega_j)} dx \tag{19}$$

In case of Gaussian pdf, it becomes:

$$d_{ij} = \frac{1}{2} Trace\left(\Sigma_i^{-1}\Sigma_j + \Sigma_j^{-1}\Sigma_i - 2I\right) + \frac{1}{2}\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right)^T \left(\Sigma_i^{-1} + \Sigma_j\right)\left(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right) \tag{20}$$

Finally, the average class separability or average divergence can be computed using the average divergence as follow:

$$d = \sum_{i=1}^{c}\sum_{j=1}^{c} \Pr(\omega_i)\Pr(\omega_j) d_{ij} \tag{21}$$

where c is the number of classes. After implementing CSM analyzes to dataset, the obtained results can be found in Table 1. For both datasets1 and dataset2, it is clearly shown that the best separable classes are first and second. Because of that, difference between means of absolute classes are greater. The condition of good separability performance is that their means of Gaussian distribution should be far from each other. In addition, in case of covariances of classes are arbitrary (uncorrelated), the performance of CSM increases.

Table 1. Class Separability Measures For Multi-Dimensional Gaussian Distributions

Datasets	**Between Classes**		
	d_{12}	***d_{13}***	***d_{23}***
Dataset 1 (2-dimensional)	107.93	49.54	9.69
Dataset 2 (3-dimensional)	4835	477.34	185.58

5. Classification of Datasets

In this section, some classification methods which are used in this study such as Minimum Risk Bayes Classification (MRBC) and Discriminant Classifier (Linear Discriminant Analysis) are explained in details. Then, the performance of these classifiers are analyzed on 2D and 3D implemented EM algorithm GMM datasets and also obtained results have shown in Table 2.

Table 2. Classification Average Accuracies Based On Implemented Expectation-Maximization Algorithm To Multi-Dimensional Gaussian Mixture Model

Datasets	**Type of Classifier**		
	Bayesian Classifier	***Discriminant Classifier (LDA)***	***Support Vector Machine (SVM)***
Dataset 1 (2-dimensional)	98.67%	98.2%	99.4%
Dataset 2 (3-dimensional)	73.88%	67.66%	78.5%

In addition, effect of amount of training and test data on accuracy of classifiers are compared as well as obtained results have shown in figures 7 and 8.

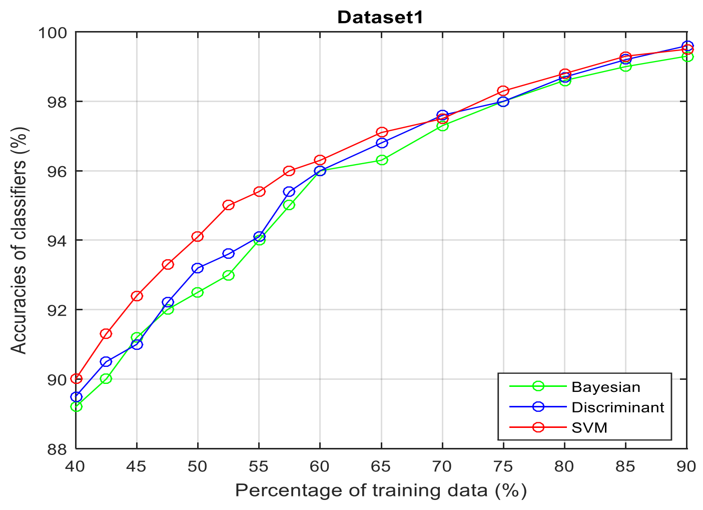

Figure 7. Influence of amount of training data on classifier accuracies

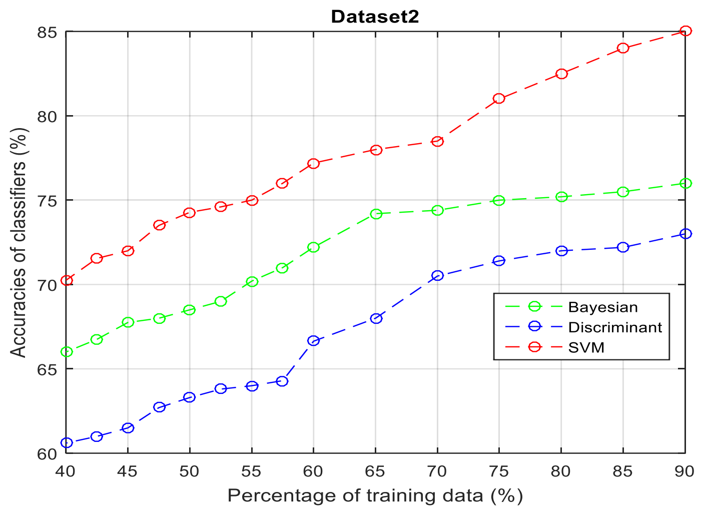

Figure 8. Influence of amount of training data on classifier accuracies

Especially, for k-NN classification, the effect of *k* parameter on classifier performance is analyzed. The obtained results have shown in Table 3.

Table 3. Average Accuracies Of K-Nn Clasifier With Various K Values Based On Implemented Expectation-Maximization Algorithm To Multi-Dimensional Gaussian Mixture Model

Datasets	**k-NN Classifier**			
	k=3	*k=5*	*k=7*	*k=9*
Dataset 1 (2-dimensional)	95.6%	94.4%	95.2%	96.4%
Dataset 2 (3-dimensional)	92%	92.66%	90.66%	91.66%

5.1. Minimum Risk Bayesian Classification

A Bayesian classifier is based on Bayes rule and the main idea is that role of a natural class is to predict the values of features for members of that class. Basically, this classifier calculates pdfs of each classes by taking mean vectors and covariance matrices for Gaussian distributions. Furthermore, the prior probabilities of the classes are taken into consideration when this operation is carried out.

For multi-classes cases, total risk or associated cost with ω_k expressed as follow:

$$r_k = \sum_{i=1}^{c} \lambda_{ki} \int_{R_i} p(\mathbf{x} \mid \omega_k) dx \tag{22}$$

where $k=1,2,..,c$ and λ_{ki} is a symbolic phenomenon and called as cost function.

$$\lambda_{ki} = \begin{cases} 0 & k = i \\ 1 & k \neq i \end{cases} \tag{23}$$

If the concerning class error costs are equal, it becomes *0* or *1* and it is called as hard decision function which is represented in (23).

In case of error costs of inter-classes are not equal, cost function transforms to another form. Thus, minimum error classification criteria is defined and the minimum the average risk is calculated with the help of r_k and cost function λ_{ki} as shown below:

$$r = \sum_{k=1}^{c} r_k \Pr(\omega_k) = \sum_{k=1}^{c} \Pr(\omega_k) \sum_{i=1}^{n} \lambda_{ki} \int_{R_i} p(\mathbf{x} \mid \omega_k) dx$$

$$= \sum_{i=1}^{c} \sum_{k=1}^{c} \lambda_{ki}\, p(\mathbf{x} \mid \omega_k) \Pr(\omega_k) \tag{24}$$

$$\ell = \sum_{k=1}^{c} \lambda_{ki} \Pr(\omega_k \mid \mathbf{x}) \tag{25}$$

(25) is rewritten in matrix-vector form as in (26):

$$\begin{bmatrix} \ell_1 \\ \vdots \\ \ell_c \end{bmatrix} = \begin{bmatrix} \lambda_{11} & \cdots & \lambda_{c1} \\ \vdots & \ddots & \vdots \\ \lambda_{1c} & \cdots & \lambda_{cc} \end{bmatrix} \begin{bmatrix} \Pr(\omega_1 \mid \mathbf{x}) \\ \vdots \\ \Pr(\omega_c \mid \mathbf{x}) \end{bmatrix} \tag{26}$$

The aim of this process is to minimize ℓ_i and assign $\boldsymbol{x}$ vector to proper class which offers minimum risk. To minimize ℓ_i, the statistical term $Pr(\omega i|x)$ should be maximized. The function of true classifying is expressed as follow:

$$\omega_{opt} = \begin{cases} \mathbf{x} \to \omega_1 & \ell_1 < \max(\ell_2, \ldots \ell_c) \\ \vdots & \vdots \\ \mathbf{x} \to \omega_c & \ell_c < \max(\ell_{c-1}, \ldots \ell_1) \end{cases} \tag{27}$$

Note that, the assignment is made with respect to the minimum cost (error). For instance, in (27), $\boldsymbol{x}$ feature vector is assigned to ω_c when the case is $\lambda_{1c}Pr(\omega_c|\boldsymbol{x}) < max(\lambda_{1c-1}Pr(\omega_{c-1}|\boldsymbol{x}),..., \lambda_{11}Pr(\omega_1|x))$. In special case prior probability of each classes are equal, the classification decision can be given by checking $Pr(\boldsymbol{x}|\omega_i)$.

5.2. k-NN Classification

k-NN classification is the one of the non-parametric density estimation methods. This method does not interest in the distribution parameters unlike to MRBC or other parametric methods. The purpose of this method is to calculate density function using k nearest parameters of neighbors as follow:

$$p(x \mid \omega_i) = \frac{k_N / N}{N_i} \tag{28}$$

where $i=1,2,...,c$ and c denotes the number of classes. Also k is a constant (must be chosen odd) and denotes the number of nearest neighbors that will be processed. N_i expresses the number of feature vector in *ith* classes. Note that, different distance metrics cause changes in classification performance. Although, different approaches can be used as distance metrics such as city-block (L_1), sum of squared difference (SSD) or Minkowski (L_∞) distance, in general Euclidean distance (L_2) is often preferred by researchers [9].

General form of sum of squared distance (SSD) is expressed in (29) as follow:

$$d_{SSD} : (x, y) \to \| x - y \|_2^2 = \langle x - y, x - y \rangle = \sum_{i=1}^{N} (x_i - y_i)^2 \tag{29}$$

and Euclidean distance can be denoted as:

$$d_2 : (x, y) \to \| x - y \|_2 = L_2(x, y) = \sqrt{d_{SSD}} = \left(\sum_{i=1}^{N} | x_i - y_i | \right)^{1/2} \tag{30}$$

and finally optimum classification decision becomes:

$$C_{opt} = \max\left(C_1, C_2, ..., C_m\right) \tag{31}$$

In (31), m is the number of classes where belongs to each k-NN vector. During classification, the distance between x vector and k nearest neighbor features are calculated and which classes belong to each vectors are also determined. In decision stage, x is marked to class C_{opt} which have the highest number of member with k features.

5.3. Discriminant Classification (LDA)

This method consists of two major title that are projection of data and determining of decision boundaries. Actually, the main objective of this method is to perform dimensional reduction. In this classification scheme, data is divided into sub-spaces or another hyperplanes, which have best class discriminatory information by the help of produced projection vectors. Hence, the best classifier performance is obtained at optimal decision boundaries. Also, during this process, it is proposed to preserve the class information as much as possible. Because of these purposes, in this subtitle, “what are the best projection vectors for discriminant classifiers” is explained in term of discriminant classifier background.

For c-classes cases, mean and covariance matrices are calculated with ignoring membership weights. Then, global mean is calculated as follow:

$$\boldsymbol{\mu}_d = \frac{1}{c} \sum_{i=1}^{c} \boldsymbol{\mu}_i \tag{32}$$

In (32), $\boldsymbol{\mu}_d$ denotes the overall mean of all data and S_w is called as With-in classes scatter matrix. This matrix is calculated with sum of the covariance matrices as illustrated below:

$$S_w = S_1 + S_2 + ... + S_c \tag{33}$$

S_B corresponds to Between-classes scatter matrix as calculated below:

$$S_{Bi} = N_i (\boldsymbol{\mu}_i - \boldsymbol{\mu}_d)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_d)^T \tag{34}$$

In (34), N_i denotes the number of samples in *ith* classes.

$$j(\omega) = \frac{\omega^T S_B \omega}{\omega^T S_w \omega} \tag{35}$$

where ω is default weight vector. In order to maximize $j(\omega)$:

$$\max(j(\omega)) = \arg\max_{\omega}\left(\frac{\omega^T S_B \omega}{\omega^T S_w \omega}\right) \tag{36}$$

Taking derivative of (35) with respect to ω and setting it zero, we obtain:

$$\frac{\partial}{\partial\omega} j(w) = S_w \omega \frac{\omega^T S_B \omega}{\omega^T S_w \omega} \tag{37}$$

If (37) is rewritten after some derivation steps, it becomes eigenvalue-eigenvector problem as shown in (38):

$$\lambda\omega = \left(S_w^{-1} S_B\right)\omega \tag{38}$$

where $\lambda = j(\omega)$ is scaler. In order to compute LDA projection vectors, by maximization process, we obtain:

$$\lambda = \arg\max_{\omega} | S_w^{-1} S_B - \lambda I | \tag{39}$$

When (38) is rewritten in matrix-vector form, *c-1* projection vectors become:

$$\begin{bmatrix} \left(S_w^{-1}S_B\right)_{1,1} & \cdots & \left(S_w^{-1}S_B\right)_{1,c-1} \\ \vdots & \ddots & \vdots \\ \left(S_w^{-1}S_B\right)_{c-1,1} & \cdots & \left(S_w^{-1}S_B\right)_{c-1,c-1} \end{bmatrix} \omega_i = \lambda_i \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_{c-1} \end{bmatrix} \tag{40}$$

Considering that, number of projection vectors depend on number of classes in GMM datasets. LDA produces *c-1* hyperplanes. The above mentioned classification methods are implemented to figures 1 and 2. The obtained results in terms of influence of amount of training and test data on classifier accuracies have shown in tables II-III and figures 7-8.

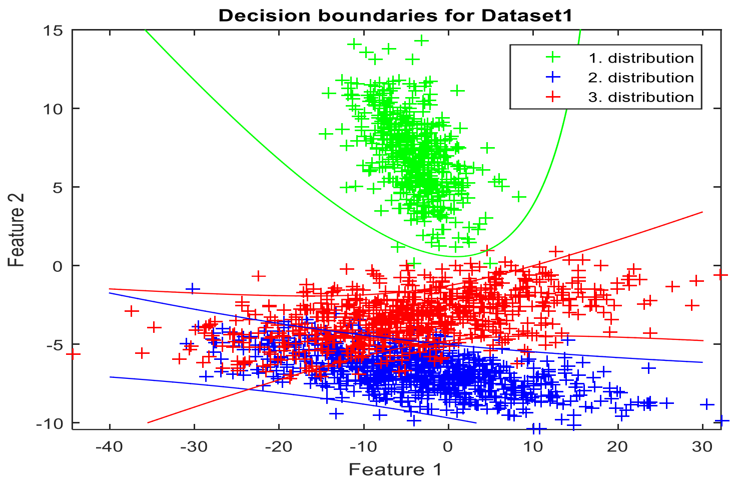

Figure 9. Decision Bounderies

In case of $\sum_i$ is arbitrary, decision boundary function of *i*th class:

$$g_i(\mathbf{x}) = \mathbf{x}^T W_i^T \mathbf{x} + \omega_i \mathbf{x} + \omega_{i0} \tag{41}$$

where ω_i is weight vector and ω_{i0} denotes bias or threshold for *i*th class. After some simplifications, the coefficient of quadratic term becomes:

$$W_i = -\frac{1}{2}\Sigma_i^{-1} \tag{42}$$

and weight vector:

$$\omega_i = \Sigma_i^{-1}\boldsymbol{\mu}_i \tag{43}$$

and finally threshold of *i*th class:

$$\omega_{i0} = -\frac{1}{2}\boldsymbol{\mu}_i^T\Sigma_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\log|\Sigma_i| + \log(\Pr(\omega_i)) \tag{44}$$

While calculating decision boundaries, for each classes, *gi(x)-g2(x)=0* hyperplanes are plotted. In this case, assuming that all data consists of two classes, it means that *g2(x)* is processed with participation of all data only except *ith* class. As a result, decision boundaries are calculated for dataset1. The obtained results are illustrated in Fig. 9.

6. Conclusion

In this paper, both dataset1 and dataset2 are modelled with GMM and EM algorithm is implemented in order to predict distribution parameters as well as estimate number of feature vectors for each classes. After that, performances of classifiers are discussed and compared. Also, in CSM analyzes for both datasets, best separable classes which are class-1 and class-2 are evaluated, hence they have arbitrary covariance values and their means are farther away from each other. In addition, to investigate the classifier performance further, k-NN classification method is implemented. It is observed that k-NN classification obtains maximum accuracy when *k=9* and *k=5* respectively for dataset1 and dataset2.

Consequently, according to all evaluation results, the best overall accuracy is observed in k-NN classification for dataset2 and the best accuracy is achieved with SVM classification for dataset1.

Furthermore, it is also determined that the classification performance decreases when dimension of space increases. Finally, when the effect of amount of training data is investigated on classifier performance, regardless of the type of classifiers it is observed that, the performance of all classifiers decrease when amount of training data increases.

References

[1] Duda R. O., Hart P.E., and Stork D.G. (2012). Pattern classification. *John Wiley & Sons*.

[2] Mahantesh K., Manjunath Aradhya V.N., and Naveena C. (2014). An impact of PCA-mixture models and different similarity distance measure techniques to identify latent image features for object categorization. *Advances in Signal Processing and Intelligent Recognition Systems*, Springer International Publishing, 371-378.

[3] Qin, Yichen Q., and Carey E. P. (2013). Maximum Lq-Likelihood Estimation via the Expectation-Maximization Algorithm: A Robust Estimation of Mixture Models. *Journal of the American Statistical Association*, 914-928.

[4] Fernández-Michelli J. I., et al. (2016). Unsupervised classification algorithm based on EM method for polarimetric SAR images. ISPRS *Journal of Photogrammetry and Remote Sensing*, 117, 56-65.

[5] Theodoridis S., et al. (2010). Introduction to pattern recognition: a matlab approach. *Academic Press*.

[6] Nguyen H. D., McLachlan G.J. (2015). Maximum likelihood estimation of Gaussian mixture models without matrix operations. *Advances in Data Analysis and Classification*, 9(4), 371-394.

[7] Bowei Y., Mingzhang Y., and Purnamrita S. (2017). Statistical Convergence Analysis of Gradient EM on General Gaussian Mixture Models. *arXiv preprint arXiv:1705.08530*.

[8] Fanglin G., et al. (2016). An expectation-maximization algorithm for blind separation of noisy mixtures using Gaussian mixture model. *Circuits, Systems, and Signal Processing*, 1-30.

[9] Li-Yu H., et al. (2016). The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus 5.1* 1304.