Applied Machine Learning
Report on Practice Assignment

TRINH Christophe - VIOLA Mario

December 16, 2016

1 Introduction

The purpose of this assignment is to apply different Machine Learning algorithms, seen during the lecture, on real-world data. Throughout this report, we explore the influence of hyper parameters of each algorithm. This work aims to have a better understanding on how to deal with these hyper parameters in order to obtain better score and performance. The overall algorithms are performed on `MLDEMOS` software.

2 Dataset

The first dataset consists of a portrait of one person. In some portraits, the person was worn glasses. Thus, we divided the dataset into two different classes: portraits with (figure 1b) and without (figure 1a) glasses (30 samples per class). Images were taken using an iPhone 6S located directly in front of the face and they had a resolution of 1932x1932. Therefore each image has more than 3 million dimensions. Additionally, pictures were taken with the same background in order to set a reference for all the dataset. The dataset also contains 3 subclasses which described the level of happiness of the person as shown in





(a) **Class 1 : Portrait without glasses** (b) **Class 2 : Portrait with glasses** (c) **subclass a : Happy** (d) **subclass b : Neutral** (e) **subclass c : Sad**

Figure 1: Image examples of each class and subclass

figure 1c, 1d, 1e. Thus, the second dataset was made based on these subclasses. 20 samples were taken for each subclass. The scale of happiness was in the range of [0:1]. 0 corresponds to a very sad humor, whereas 1 very happy. This dataset will be used for the section 6.

3 PCA

The goal of PCA is to project our dataset into a new compact base which keeps enough variance of data (i.e. information). Processing high-dimensional data requires lot of computational resources. Thus, we need to reduce the dimension of data using PCA in order to extract the main features. Firstly, we load the dataset 1 into `MLDEMOS` and split the two different classes. On figure 2, we show the eigen value of each component projected and the reconstruction error function of eigen vectors. We decided to keep approximately 80% of variance, therefore the 12 first eigen vector are used to compute the PCA.

After applying the PCA on our dataset, we obtain several eigenvectors, we will introduce two of them with a manual graphical classification. Firstly, the primary eigenvectors, figure 3a, shows a well separation between the two classes with no overlapping between samples. Contrary to the first scatterplot, the separation of the classes in the second one, figure 3b, is not very accurate. We can observe an overlap; some of the data points in class 1 are inside the cluster of class 2, thus, this scatterplot doesn't represent efficiently our classes for a classification step. The eigenvalue of the primary plan is 45,3 % comparing to 34,8 % for the second one. That means the primary plan contains more information than the second one. Therefore, we decide to choose the primary projection to perform the classification.

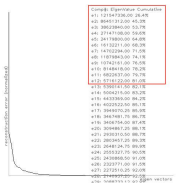
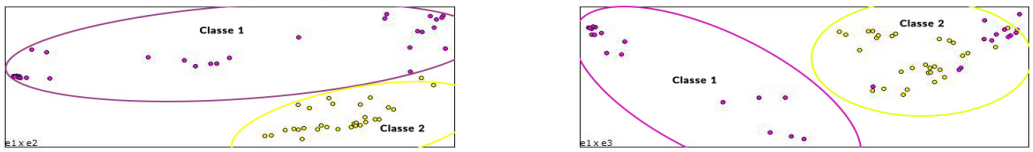


Figure 2: Eigen value cumulative and reconstruction error



(a) **Scatterplot with eigenvector 1 x 2** (b) **Scatterplot with eigenvector 1 x 3**

Figure 3: Separation of the classes with PCA

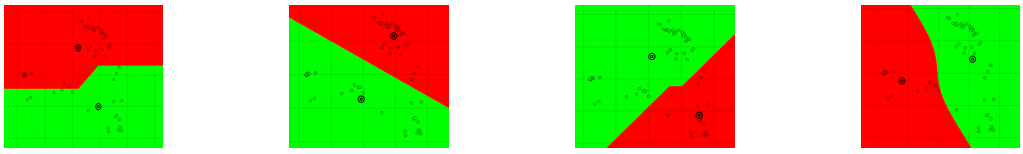
4 Clustering

In this section, we will use three different clustering techniques and assess the performance of each by changing the hyperparameters. We tried to optimize the hyperparameters regardless of different measures, internal and external.

4.1 Qualitative assessment

4.1.1 K-means method

In this method, the only hyperparameter is the number of cluster K. K-means consists on computing the distance between the data points and the centroid to perform clustering. The initial location of all centroids is different each times we run the algorithm. The choice of the distance has an impact on the results. As the first step, we apply this clustering technique to our first dataset (on a specific PCA plane : 1x2 figure 3a) with several different distances, the result is shown on figure 4. We note the borders are different for each distance. The *Manhattan distance* refers to the sum of distances along each dimension (horizontal and vertical) that is why the cluster border is the longest comparing to the others distances. While using the *infinity metric*, only one of the coordinates are required to compute it. Thus the borders of the clusters are segmented when the coordinate increases more than the others. By using the *polynomial k-means*, the distances are calculated using exponents. Thus the borders of the clusters are curved instead of straight lines. The higher the exponent is, the smoother are the borders.



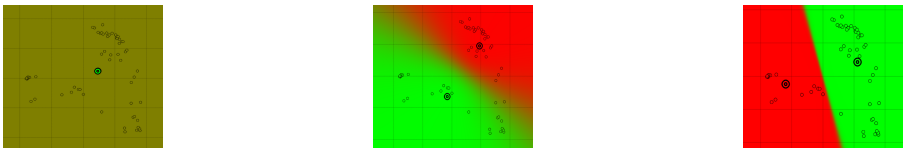
(a) **Manhattan** (b) **Euclidean** (c) **Infinity** (d) **Polynomial**

Figure 4: K-means clustering method with different metrics

4.1.2 Soft K-means method

The soft K-means technique has an another hyperparameter called β, it represents the stiffness of the borders. This parameter is also related to the disparity across clusters (inversely proportional to β). The responsibility of each sample is a real number varying between 0 and 1. For each iteration of the algorithm, the centroids are adjusted to match the weighted means of the data points that they are responsible for. In other words, the sample which are between two clusters will have less responsibility

in the soft K-means method than hard K-means method.
In our case, we found that a value of 10 for β gives a fairly good clustering. In fact, with a β of 4, the disparity in the cluster is high. Consequently, the algorithm found only 1 cluster. Furthermore, the higher the β value is, the more the Soft-K-Means results looks like the standard method as we can see on 5 with a β equals to 100.

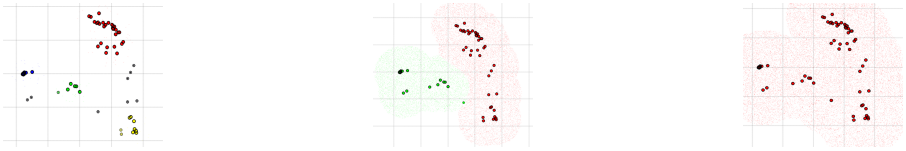


(a) $\boldsymbol{\beta = 4}$ (b) $\boldsymbol{\beta = 10}$ (c) $\boldsymbol{\beta = 100}$

Figure 5: Soft K-means clustering method with different values of β

4.1.3 DBSCAN method

In this method, there are 2 hyperparameters ϵ and m_{data}. The variable ϵ characterized the size of the neighborhood and m_{data} the minimum number of data points required to consider the group of points as a cluster. If the size of neighborhood ϵ is too high, the algorithm may merged different closed clusters into one. Thus, the risk is to have less cluster than expected if the data points are not well separated. However, if the ϵ is too low, the algorithm may detect more outliers and more clusters.Hence, data points won't be well clustered as we can see on figure 6. In our case, we found that a value of 0.230 for ϵ, results in a satisfactory clustering.



(a) $\boldsymbol{\epsilon = 0.100}$ (b) $\boldsymbol{\epsilon = 0.230}$ (c) $\boldsymbol{\epsilon = 0.300}$

Figure 6: Influence of ϵ parameter on DBSCAN method

4.2 Quantitative assessment

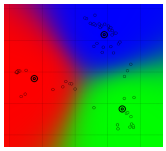
In this section, we will perform K-Means and Soft-K-Means clustering methods and looking at different metrics. If we optimized the algorithm regarding the RSS, we always found a large number of cluster as a result as we can see on figure 7. The higher is the number of cluster K, the lower is the RSS value. Thus, it is not relevant to optimize regarding the RSS because at the end, it will have more cluster than expected.

(a) **Soft K-means ($\boldsymbol{\beta = 10}$ - range [0,8])** (b) **K-means (norm L2 - range [0,8])**

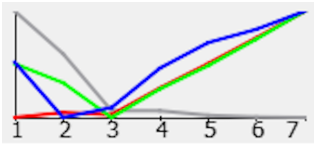
Figure 7: Results of optimization regarding RSS

Once we optimized the algorithm with respect to a specific metric (BIC-AIC), we generally get 3 clusters as we can see on the figure 8. It can be explained by the distribution of the points which are sparse.

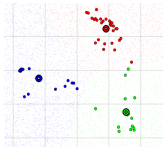
Yet, optimizing regarding to the BIC allowed to get 2 clusters but the hyperparameter has to be well-tuned as we can see on the figure 9 especially in the case of Soft K-means with β. In our case,



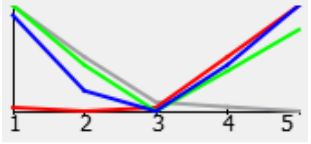
(a) **Soft K-means - β = 10**



(b) **Curve of optimization - Soft K-means $\beta = 10$. (RSS, BIC, AIC and F1-Measure)**



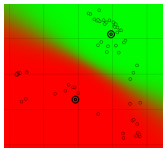
(c) **K-means - norm L2**



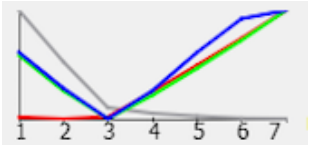
(d) **Curve of optimization - K-means norm L2. (RSS, BIC, AIC and F1-Measure)**

Figure 8: Results of optimization regarding AIC

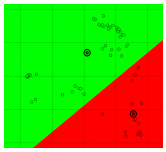
penalizing the likelihood with the number of free parameters with the number of data points (BIC) give results more favorable than penalizing only with the number of free parameter (AIC).



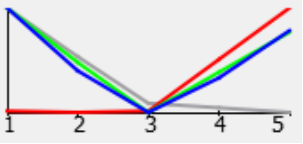
(a) **Soft K-means - β = 10**



(b) **Curve of optimization - Soft K-means $\beta = 20$. (RSS, BIC, AIC and F1-Measure)**



(c) **K-means - norm L2**



(d) **Curve of optimization - K-means norm L2. (RSS, BIC, AIC and F1-Measure)**

Figure 9: Results of optimization regarding BIC

In this part, we will evaluate the semi-supervised learning on our dataset. We vary the percentage of labeled data going from very low ratio to high ratios and optimize with respect to F1-measure.

With the soft-K-means method and 10% of the data labeled, we obtained 3 clusters (10a) whereas we expected only 2 but the F1-measure is equal to 0.90, close to 1. If we increased the ratio to 50% the algorithm is able to separate the data into 2 clusters (10c) and we have a good F1-measure of 0.91.

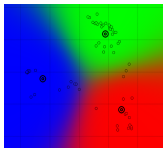
With the standard k-means method, in our case, the algorithm is not able to well-clustered the data points (11a) with 50% of the dataset labeled. It needs 100% to get 2 clusters (11c) at the end. Thus, this algorithm is not relevant in this problem because it costs a lot in terms of data labeled.

5 Classification

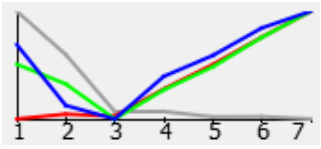
In this section, we will quantify the performance of different methods of classification in a quantitative manner using the F-measure. We will also compare the sensitivity of the algorithm's performance regarding to the choice of hyper-parameters and the splitting ratio between training and testing set.

5.1 GMM

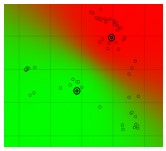
GMM and Bayes technique depends on different hyper-parameters such as the covariance matrix or the number of components per class. We will evaluate quantitatively the impact of each parameters on the



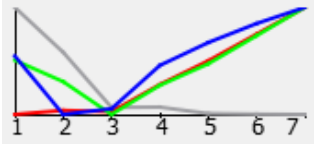
(a) $\beta = 10$, **10% labeled**



(b) **Curve of optimization - $\beta = 10$ - 10% labeled. (RSS, BIC, AIC and F1-Measure)**

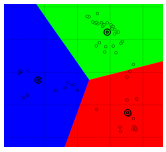


(c) $\beta = 10$, **50% labeled**

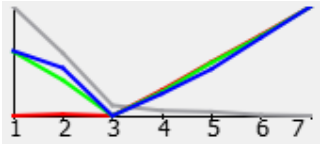


(d) **Curve of optimization - $\beta = 10$ - 50% labeled. (RSS, BIC, AIC and F1-Measure)**

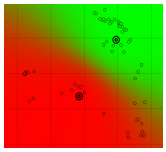
Figure 10: Soft K-means - results of optimization regarding F1



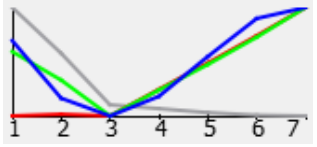
(a) **L2 norm - 50% labeled**



(b) **Curve of optimization - L2 norm - 50% labeled. (RSS, BIC, AIC and F1-Measure)**



(c) **L2 norm - 100% labeled**



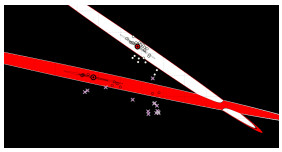
(d) **Curve of optimization - L2 norm - 100% labeled. (RSS, BIC, AIC and F1-Measure)**

Figure 11: **K-means - results of optimization regarding F1**

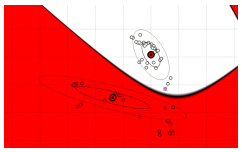
performance and we will compare the F-measure for a set of parameters to tackle the issue of over-fitting.

Firstly, we have evaluated the effect of number fold in the cross-validation. In `MLDEMOS`, the number of fold seems to be the number of times we trained the model. In fact, we can tune the training/test ratio and we can see that with a low number of folds such as 2, the F1-measure is not representative. But with a number of folds of 10, the results are coherent. We decided to keep a 10-fold for the following.

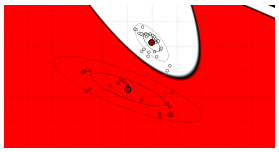
Secondly, we have evaluated the effect of the training/testing ratio on the performance. We kept the same parameter to compare the performance. We set a full covariance matrix with 1 component per class. Results are shown in figure 12



(a) **Train ratio = 10%**



(b) **Train ratio = 50%**



(c) **Train ratio = 75%**

Figure 12: GMM - Full covariance - 1 GMM per class

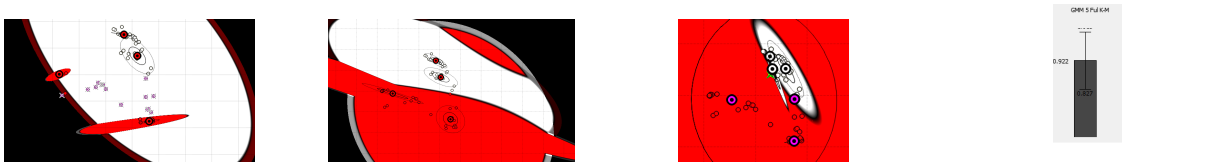
As we have 60 samples in our dataset, with a training ratio of 10%, only 6 samples are used for training. Thus, the trained model leads to bad performance. It's a case of under-fitting. However while increasing the training ratio, the model is able to correctly classify the data. Yet, with a too high training ratio, the model may over-fit on the data as we can see on figure 13 with a 10-fold. In fact, on training

set, the variance is very low but on the testing set, the variance of the F1-measure is very high. Thus, it's a case of over-fitting.

(a) **F-measure on testing set** (b) **F-measure on training set**

Figure 13: GMM- Full covariance - 1 GMM per class - Train ratio = 75%

Thirdly, we have evaluated the effect of number component per class. We used a full-covariance matrix. We've seen previously in figure 12, with one component per class, the model needs at least 50% of training set. While increasing the number of component, the model gives better performances with less training data as we can see on figure 14. In fact, with 2 components and a ratio of 25%, the model gives bad classification. It needs 50% of training set to correctly classify the data without over-fitting but it is more costly. With 5 components per class, the model needs 50% less of training data to correctly classify the data but the model is over-fitting. In fact, the variance of the F-measure on testing set in this case is very high 14d whereas on training set very low.



(a) **2 components per class - 25% train ratio**

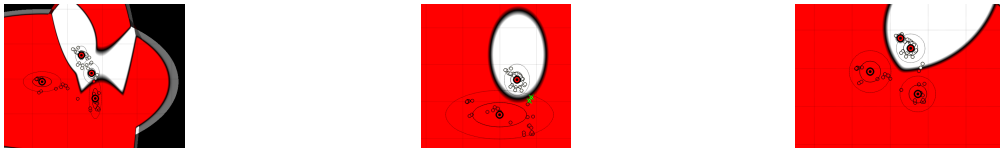
(b) **2 components per class - 50% train ratio**

(c) **5 components per class - 25% train ratio**

(d) **F-measure on testing set - 25% train ratio - 5 components**

Figure 14: GMM - Full covariance

Then, influence of covariance matrix is evaluated. We've seen previously that with a full covariance matrix and 2 components per class, we needs 50% of training data. With a diagonal matrix and only 1 component per class, even if we used 100% of training data, the model is not able to classify perfectly the data. But with 2 components per class and a training ratio of 33%, the diagonal matrix model is able to perfectly classify the data without over-fitting. And for the spherical matrix, it needs 2 components but 50% of training data. Results are shown in figure 15.



(a) **GMM Diagonal Covariance - 33% - 2 Components**

(b) **GMM Diagonal Covariance - 100% - 1 Component**

(c) **GMM Spherical Covariance - 50% - 2 Components**

Figure 15: Influence of covariance matrix

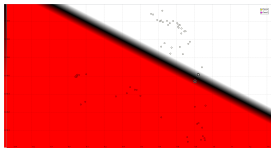
5.2 SVM

In this section, we will consider two different Support Vector Machine algorithm. The first one performs linear classification and consist of one hyper-parameter which is C-Value. The second algorithm RBF kernel perform a non linear classification with one more hyper-parameter : the kernel width. The C-Value controls the influence of each individual support vector. A large C-Value will penalize the cost

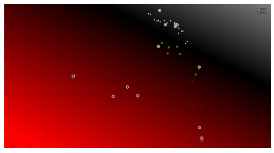
of misclassification dataset a lot. The best C-value is a trade-off between maximizing the margin and minimizing the classification errors. The kernel width defines the width of the kernel function. Large kernel width would imply a smoother fit and thus having a larger influence of neighborhood data point and vice-versa.

5.2.1 Linear

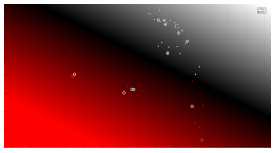
As we can observe on figure 16, the resulting classification for different C-value and fixed train ratio (25%). We note for the C-value : 0.5, there are 15 support vectors and 8 points misclassified. Then with higher C-value, we penalize the cost of misclassified data points in order to reduce them. The figure 16d has still one misclassified points and 8 support vectors. The misclassified point is close to the other class, we can increase C-value and train ratio in order to reduce misclassified datapoints. Nevertheless, with a high C-value, the cost of misclassified datapoint will be high and the margin will reduce, thus forcing the algorithm to explain the data stricter and potentially overfit. The goal is to find a trade off between maximizing the margin and minimizing the classification errors. We can see an over-fitting case on figure 16a, which consist of only two support vectors and have a very low margin. Therefore, this model has a poor generalization.



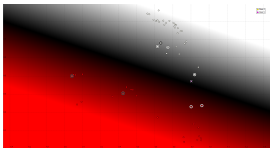
(a) **C-value : 1000 and train ratio : 50%**



(b) **C-value : 0.5 and train ratio : 25%**



(c) **C-value : 1 and train ratio : 25%**



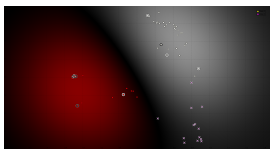
(d) **C-value : 5 and train ratio : 25%**

Figure 16: SVM linear : influence of C-value

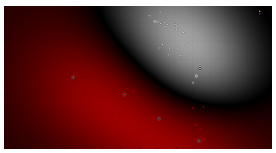
5.2.2 RBF

In our case, the dataset is linearly separable thus the linear SVM algorithm is sufficient. However, we perform RBF SVM algorithm to understand the influence of the hyper-parameters.

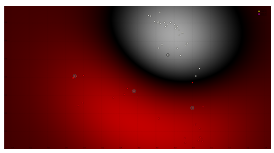
We observe on the figure 17a , the kernel width chosen is 0.2 and it seems to be the right size oh the kernel but we obtain 13 misclassified data points. This is due to the support vectors chosen by the algorithm. In order to perform a better model, we increase the C-Value to decrease the number of misclassified data points. On figure 17b , it has one misclassified data point, nevertheless the classification has a good generalization. If we continue to increase the C-Value, as we observe on figure 17c , the classification has none misclassified data point but it seems to overfit the dataset. Finally, we try with a higher kernel width, as shown on figure 17d , the resulting classification has a good generalization and only one misclassified data point comparing to the figure 17a , which has the same C-Value. Therefore, we can obtain better classification only by increasing the kernel width.



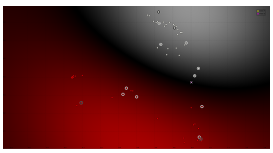
(a) **C-value : 0.5 and kernel width : 0.2**



(b) **C-value : 1 and kernel width : 0.2**



(c) **C-value : 10 and kernel width : 0.2**



(d) **C-value : 0.5 and kernel width : 0.5**

Figure 17: SVM RBF : influence of kernel width and C-value

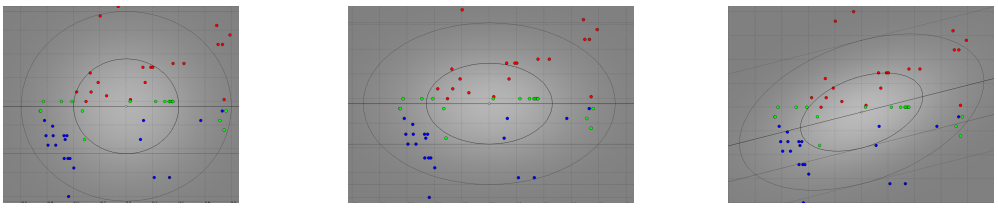
6 Regression

6.1 GMR

Firstly, we choose the best input dimension for regression. We select the one which has low overlapping value. In this section, we will performed a Gaussian Mixture Regression and changing the different parameters of this model.

6.1.1 Qualitative evaluation

Firstly, we evaluate the influence of the covariance matrice on the regression with a K-means initialization, 1 mixture component and a training ratio of 100%. Results are shown on figure 18.



(a) **Spherical** (b) **Diagonal** (c) **Full**

Figure 18: GMR : influence of covariance matrix

A spherical and diagonal covariance matrix both give a straight line which represents the mean of the datapoint. In fact, following the regressive signal obtained :

$$y = \sum_{i=1}^{K} \beta_i(x)(\mu_y^i + \Sigma_{xy}^i(\Sigma_{xx}^i)^{-1}(x - \mu_x^i)) \tag{1}$$

since for spherical and diagonal the quantity, $\Sigma_{xy}^i = 0$, the result is just the mean of the datapoint.

Then we evaluate the effect of the number of mixture components with a K-means initialization, a full covariance matrix and a training ratio of 100%. Results are shown on figure 19.

(a) **3 components** (b) **6 components**

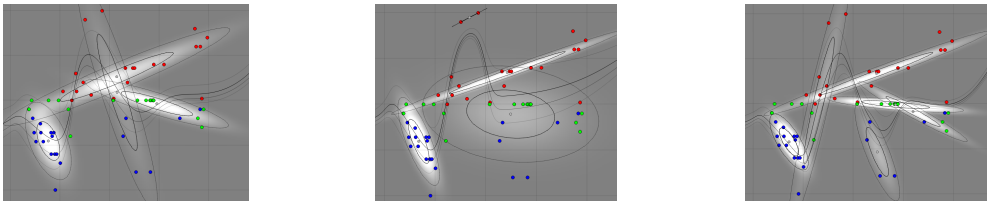
Figure 19: GMR : influence of number of mixture components

With 3 mixture components, there is a poor fit with the large gaussian on the end of the dataset. That is due to the unbalanced number of points in relation to the region and the spread of those points. A solution can be to increase the number of points. In fact, with 6 mixture components, the datapoints are better fitted.

Then, the effect of the initialization is tackled. We used a full covariance matrix and training ratio of 100%. Results are shown on figure 20.

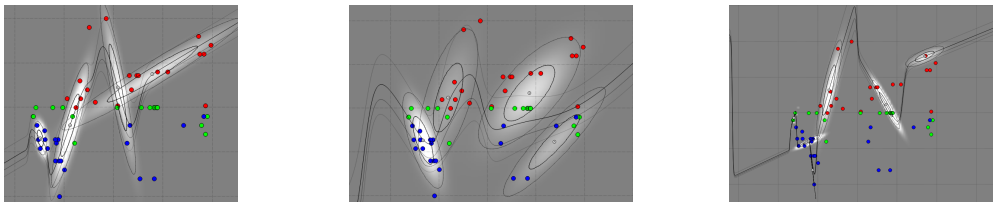
We can see that depending on the initialization used, the gaussians fit different points and give different results. For instance with a random initialization and 4 mixture components, in the top, 2 datapoints are fit by one gaussian. Hence, with only 4 mixtures components, the other 3 gaussians can't fit well the datapoint. Hence, we need to increase the number of mixture components which is more costly. Using a uniform or a k-means initialization avoid this situation.

Then, we change the training ratio using a full covariance matrix and a kmeans initialization. Results are shown on figure 21.



(a) **Uniform - 4 components** (b) **Random - 4 components** (c) **Random - 7 components**

Figure 20: GMR : influence of initialization



(a) **33% - 4 components** (b) **66% - 4 components** (c) **33% - 6 components**

Figure 21: GMR : influence of training ratio

If we used a too low number of mixture with 33% of training ratio, the model can missed some points depending on the initialization. To avoid this, we've to enhance the number of datapoint in the training set. As expected with 66% of training ratio, the model is more able to fit the data. Moreover, we can notice that while increasing the complexity of the model with a larger number of mixture components, with a low training ratio, the model is not able to fit well the datapoints.

6.1.2 Quantitative evaluation

In this section, we performed a quantitative evaluation on the all dimensions dataset. We compared different configuration of parameters and looked at the regression error in terms of MSE. We used cross-validation upon 30 folds. Results are shown in figure 22.

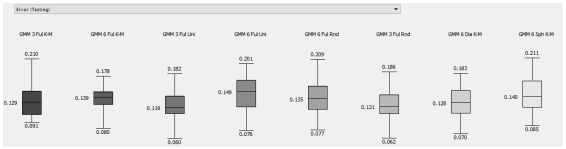


Figure 22: **Error testing - 66% Training ratio**

We have seen that in overall, while increasing the training ratio, the error on testing set decreased a little. Moreover, when we take a look the error for each configuration chosen, it seems that the difference between each model is small. We can also notice the variance is large on the figure for every model meaning that the models are reliable. However, it is very dependent on the initialization as we have seen while running many times.

6.2 SVR

Support Vector Machine can also be used as a regression method and keep the main features that characterize the algorithm. The Support Vector Regression (SVR) is a regression method and determine a regressive signal on a subset of the datapoints call support vectors. The output signal is computed with the following formula :

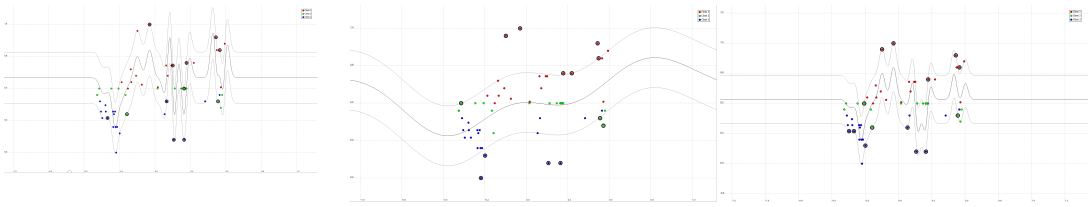
$$y = f(x) = \sum_i \alpha_i k(x^i, x) \tag{2}$$

which $k(x^i, x)$ represents the kernel function, α_i is a linear coefficient. As we seen on SVM classification section, the algorithm consist of several hyper-parameters. The C-value parameter control the penalty term on poor fit and ϵ parameter determines the minimal required precision. The predicted output will be in between the *tube* corresponding to $f(x) \pm \epsilon$.

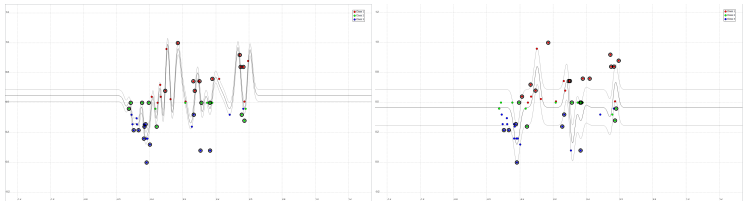
6.2.1 Qualitative evaluation

Firstly, we will perform a regression with a RBF kernel on the 2 dimensional dataset chosen. As we can see on the figure 23a , the RBF kernel width is small (0.001), thus the kernel places a Gaussian function on each support vectors with a small standard deviation that why the regression overfit the data. Now, if we increase the kernel width (0.3), as shown in the figure 23b , the Gaussian function on each support vectors is affected to a higher standard deviation. That means, each support vector will be influenced by the others support vectors neighbor. The resulting regression is smoother than the previous regression. In order to reduce the effect of the kernel width on the first regression, we reduce the penalty factor C (1.5 comparing to 10) that leads to reduce the overfitting, as we can see on figure 23c. If we reduce the size of the $\epsilon - tube$ (0.05 comparing to 0.2 for the previous regression), the resulting regression, on figure 23d, will overfit the data.

Then, we decrease the train test ratio to 33%, the figure 23e show the obtained regression. We observe the number of support vector decrease and depend on the initialization.



(a) **kernel : 0.001, C-value:10, $\epsilon - tube$: 0.2, TR: 100%** (b) **kernel : 0.3, C-value:10, $\epsilon - tube$: 0.2, TR: 100%** (c) **kernel : 0.3, C-value:1.5, $\epsilon - tube$: 0.2, TR: 100%**



(d) **kernel : 0.001, C-value:5, $\epsilon - tube$: 0.05, TR: 100%** (e) **kernel : 0.001, C-value:10, $\epsilon - tube$: 0.15, TR: 33%**

Figure 23: SVR : influence of differents parameters

6.2.2 Quantitative evaluation

We performed several regression with different hyper parameters on the (N+1) dimensions dataset, then we compare the performance of the resulting regression in terms of Mean Square Error. As shown on the figure 24, the configuration which has the less error is one with the small kernel width. However, the algorithm with higher $\epsilon - tube$ has the a small variance and a larger error, that means the regression is reliable.

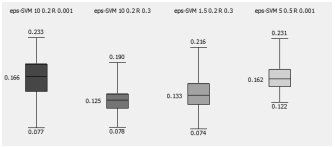


Figure 24: **Error testing - 66% Training ratio**

7 Overall discussion and conclusion

As a conclusion, during this lab we have realized the difficulty to work with real-world data especially in the case of the regression. With our dataset, the clustering algorithms seems to be less efficient than the GMM and SVM to perform classification. Moreover, regression was difficult to do with our dataset because of the complexity of our dataset (many points have the same x-axis value).