Applied Machine Learning
Report on Practice Assignment

Team 34

Fall Semester 2017

1 Introduction

This report focuses on the application of machine learning techniques, supervised and unsupervised. In particular, a linearly separable dataset consisting of images belonging to 4 different classes was constructed. PCA was then applied to reduce its dimensionality and clustering techniques such as K-Means, Soft K-Means and DBScan were used to perform unsupervised class separation. For the K-Means techniques, the effect of random initialisation was also analysed.

In a second moment, a new dataset was created, which derived from the first one but was made less linearly separable by the addition of some noisy points. This dataset was then used to test the performance of different classification techniques, such as GMM, linear and non-linear SVM. Finally, these methods were compared and the effect of parameters such as train/test ratio and number of folds for cross-validation was studied.

2 Datasets

The initial dataset was composed of 4 classes with 12 images per class. Figure 1 shows the composition of the different classes, that were chosen based on their varied geometrical and chromatic features: Watches are dark and have a round shape, bananas are long and yellow, chairs have angular features and pens are long and thin. This representative features are summarized in Table 1.

(a) Banana (b) Chair (c) Pen (d) Watch

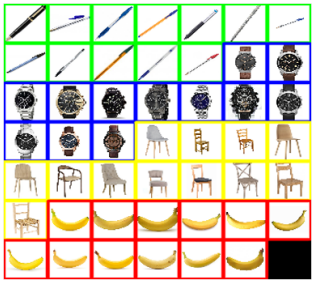
Figure 1: Four different classes

The pictures are normalized when imported in MLDemos to have the same size. The initial dataset (see Figure 2a) is stored in a single 336×336 image constituted of a 7 by 7 matrix of images. Thus, each image has a size of 48×48 pixels. With a RGB color format, there is a number of $48 \times 48 \times 3$ features.

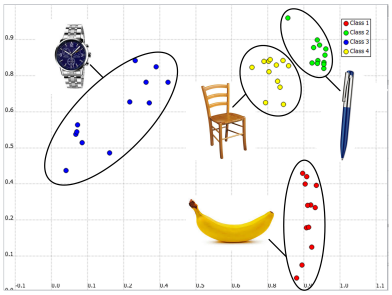
Classes	**Shape**	**Colour**	**Number of images**	**Size in pixels**
Banana	Long	Yellow	12	$48 \cdot 48$
Pens	Long and thin	Multicoloured	12	$48 \cdot 48$
Watches	Round	Dark	12	$48 \cdot 48$
Chairs	Angular	Light	12	$48 \cdot 48$

Table 1: Dataset characteristics

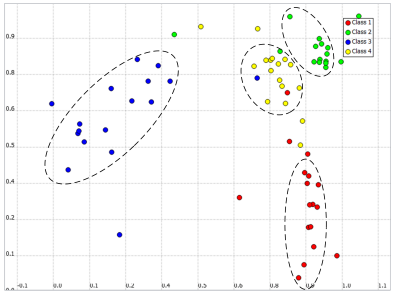
An effort was made to have a linear separable dataset in the beginning for learning purposes. In order to obtain that, the images for each class were purposely chosen to have similar caracteristics such as shape, color and orientation in space. For the second part of this report the same dataset was used, but some noisy datapoints were added by hand in MLDemos (5 per class) for a total of 17 datapoints per class. This new dataset can be seen in Figure 2c.



(a) Pictures in dataset 1



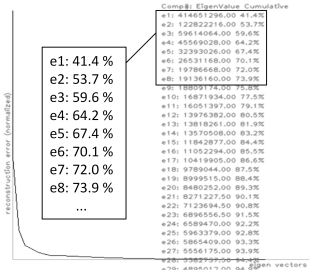
(b) Projected datapoints of dataset 1



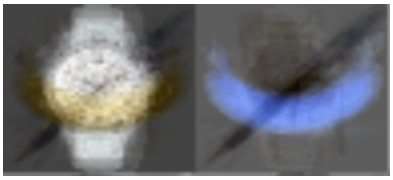
(c) Projected datapoints of dataset 2

Figure 2: Comparison of the datasets used for the report

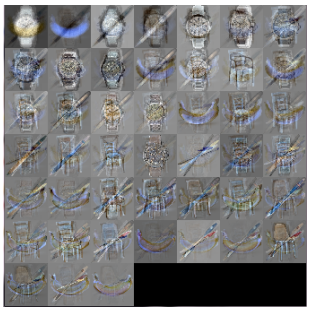
3 Dimensionality Reduction



(a) Cumulative eigenvalues



(b) First two eigenvectors



(c) All eigenvectors

Figure 3: First PCA

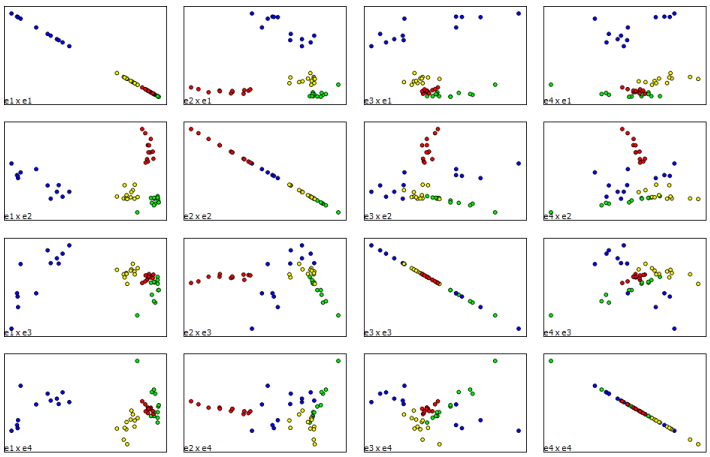


Figure 4: PCA projection matrix on eigenvectors 1 to 4

PCA was applied to the dataset 1. Figure 3a shows the cumulative reconstruction error: it can be noticed that 73.9% of the dataset can be reconstructed with the first 8 eigenvectors. It can be seen in Figure 4 that the projection onto e1-e2 was the optimal projection for clustering purposes as could be normally expected: indeed they represent most of the variance of the data. By further studying Figure 3b, the first eigenvector corresponds to a white watch, a yellow banana and the shape of a pen in the background. On the other hand the second eigenvector shows a blue banana, a pen and a shape of a chair in the background. As such, eigenvector 1 can be expected to separate well the watches class from the rest, while the second one will separate better the banana class from the rest. The two principal eigenvectors are shown in Figure 3b while the first 45 eigenvectors can be seen in Figure 3c.

4 Clustering - qualitative assessment

In order to assess the functionalities of unsupervised clustering methods, the dataset 1 was used (see Figure 2a). In particular, the algorithms K-Means, Soft K-Means and DBScan were applied.

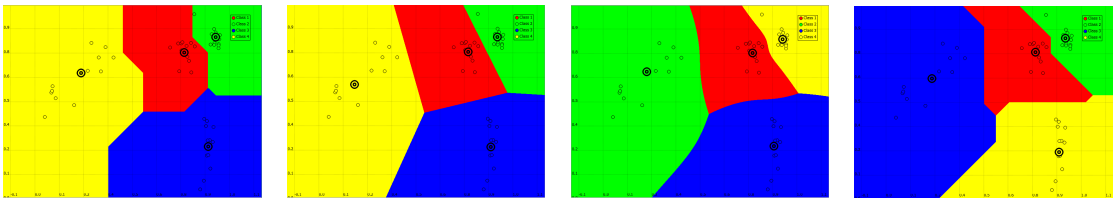
4.1 K-means

The hyperparameters for the K-means method are:

- the number of clusters;
- the metric (L_1, L_2, L_∞, L_p).

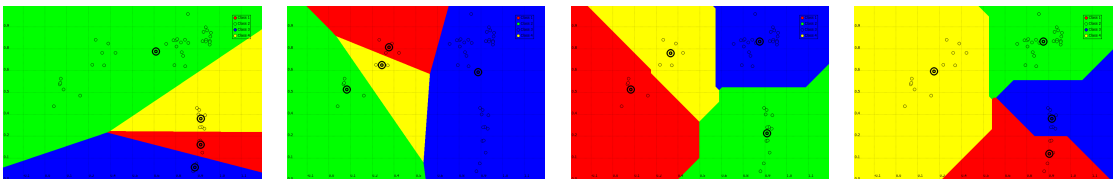
In Figure 5, correct results (when compared to the original labeled data) are shown for K-Means applied for K=4 and different metrics. It can be observed that the type of metric affects the shape of the borders between cluster. Indeed, L_1 defines distances that are the sum of the coordinates, as such the edges generated will have $n \cdot 45°$ slopes. On the other hand, L_2 defines distances in the euclidian way, as such the edges generated will be straight. Finally, L_p distances will generate higher parabolic functions, as the p degree of the polynomial function increases. When $p \to \infty$, L_∞ is defined which generated similar edges as the L_1 metric.

Although all metrics were able to correctly separate the classes, some classification errors could typically be obtained, as can be seen in Figure 6. These errors can be mainly due to the random initialisation of K-Means (see Initialization below) and the fact that the classes do not have similar distributions, which is the condition K-Mean works best for.



(a) Metric L1 (b) Metric L2 (c) Metric Lp with p=3 (d) Metric Linf

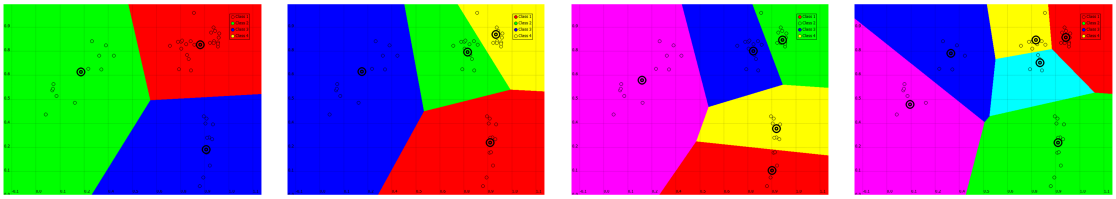
Figure 5: Dataset 1 with correct K-Means clustering, different metrics and K=4



(a) Metric L2 (b) Metric L2 (c) Metric Linf (d) Metric Linf

Figure 6: Dataset 1 with incorrect K-Means clustering, metrics L2 and Linf and K=4

Finally, K-Means behaviour was tested for different values of K (see Figure 7). Since clustering does not use the label of each point, the separation of datapoints is done according to the position of the points in the PCA projection. For dataset 1, the classes Chairs and Pens are quite close in the PCA projection (see Figure 2b). As such, when the cluster number is smaller than the total number of classes as in Figure 7a, the classes Chairs and Pens are left in the same cluster.



(a) K = 3 (b) K = 4 (c) K = 5 (d) K = 6

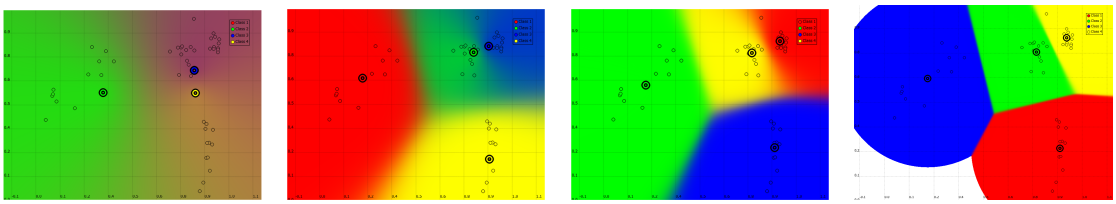
Figure 7: Dataset 1 with K-Means clustering, Euclidean metrics and different value of K

4.2 Soft K-Means

The hyperparameters for the soft K-means method are:

- the number of clusters;
- the stiffness β.

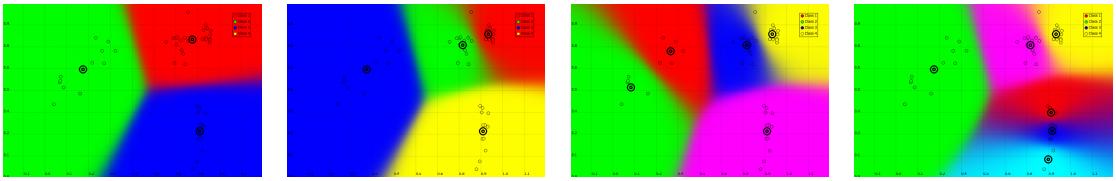
In Figure 8, results for K=4 and different values of β are shown. The higher the value of β, the more similar soft K-Means becomes to hard K-Means. When the stiffness is too low, the classes are difficult to distinguish. From $\beta = 30$, the 4 classes are clearly distinguished and correctly clustered.



(a) $\beta = 5$ (b) $\beta = 15$ (c) $\beta = 30$ (d) $\beta = 200$

Figure 8: Dataset 1 with soft K-Means clustering, K = 4 and different value of β

It can be noticed that the results for soft K-Means when varying the K don't differ considerably from what obtained with hard K-Means, as shown in Figure 9.

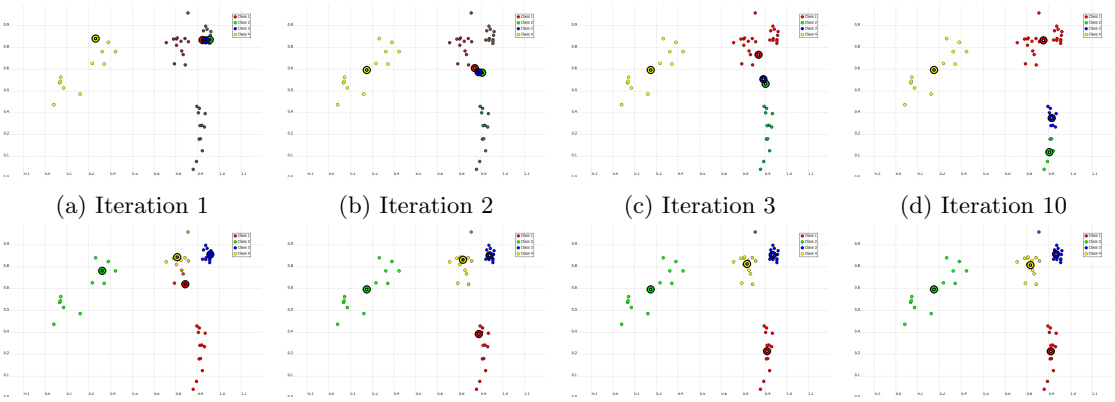


(a) K=3 (b) K=4 (c) K=5 (d) K=6

Figure 9: Dataset 1 with soft K-Means clustering,$\beta = 30$ and different values of K

Initialization

The methods above apply a random initialisation of the cluster centers. The effect of this random procedure can be observed in Figure 10. In particular, Figures 10a to 10d show an initialization that results in an incorrect clustering, while Figures 10e to 10h show a correct clustering starting from iteration 3.



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 10

(e) Iteration 1 (f) Iteration 2 (g) Iteration 3 (h) Iteration 6

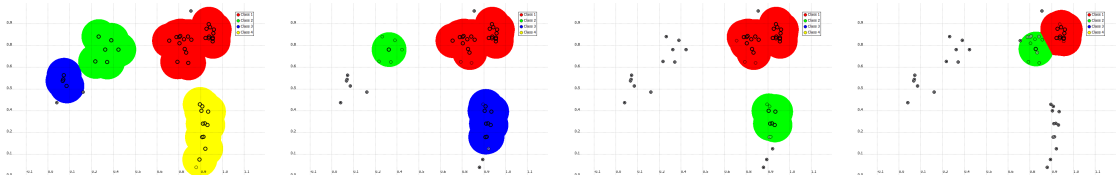
Figure 10: Dataset 1 with soft K-Means clustering for K=4 and $\beta = 30$, showing effects of random initialization

4.3 DBSCAN

The hyperparameters for the DBScan method are:

- the minimum number of points in a cluster MinPts;
- the size of neighborhood ϵ.

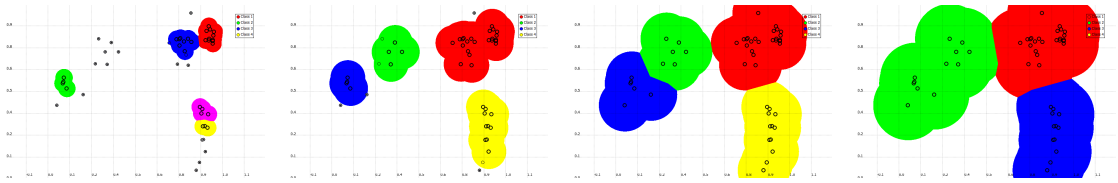
It can be noticed that when the number of Minpoints increases (see Figure 11), the smaller clusters are lost, and more and more points are considered outliers as the ϵ doesn't change.



(a) $Minpoints = 2$ (b) $Minpoints = 4$ (c) $Minpoints = 6$ (d) $Minpoints = 10$

Figure 11: Dataset 1 with DBSCAN clustering, $\epsilon = 0.1$ and different value of $Minpoints$

The value of ϵ was then varied for a fixed Minpt = 2, as it can be seen in Figure 12. For smaller values of ϵ (as Figure 12a), the size of the clusters is expectedly smaller, and as the density of the dataset is not uniform, many outliers are present. This method results in clusters than are not necessarily globular, which is an advantage compared to K-Means for this particular dataset, but it clusters datapoints whose inter-distance is smaller than ϵ. As such, in this particular case, the classes Chairs and Pens can be distinguished only for small values of ϵ, which do not cluster well the other classes as their density is lower. When the value of ϵ is higher, (i.e. Figure 12d), the classes Bananas and Watches are correctly clustered but the other two are fused togheter.



(a) $\epsilon = 0.05$ (b) $\epsilon = 0.1$ (c) $\epsilon = 0.15$ (d) $\epsilon = 0.2$

Figure 12: Dataset 1 with DBSCAN clustering, $Minpoints = 2$ and different value of ϵ

4.4 Algorithm comparison

It could be seen that K-Means and Soft K-Means yield similar results, which are quite satisfactory. On the other hand, DBScan does not perform as well, since the inhomogeneous distribution of the different classes makes it difficult to choose a meaningful combination of Minpts-ϵ that behaves well for all classes.

As for the computational cost, K-Means is the least costly, followed by Soft K-Means, while DBScan is computationnaly intense. As such, the optimal method to cluster the data in this case would be K-Means.

5 Classification - quantitative assessment

5.1 Methods comparison

The purpose of this section is to compare the GMM and SVM method with the dataset 2. Several factors can influence the performance of the GMM and C-SVM classification methods. Since there are many factors to take into account, a comparison plan is fixed in Figure 13. The first parameters to set are the test/ratio train and the number of folds for crossvalidation because these parameters directly influences the classification results for each method.

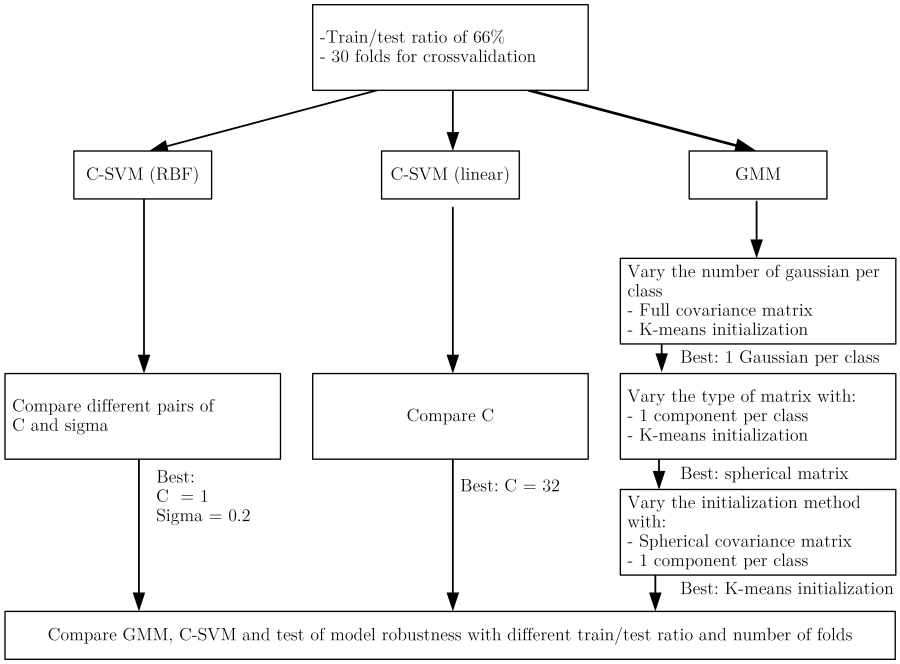


Figure 13: Comparison method block diagram

5.2 GMM classification

For the GMM classification, the hyperparameters to be found were:

- the number of gaussian components;
- the type of matrix (spherical, diagonal, full);
- the initialization method (random, uniform, K-Means).

Quantitative analysis of classification methods are done with a train/test ratio of 66% and 30 folds for crossvalidation. Further analysis on the choice of these two parameters is made in Section 5.5. Then, the hyperparameters were found in three stages. First, the K-Means initialization mode is set as it is a method expected to be more efficient than the random and uniform methods. A full covariance matrix is set to allow the algorithm to chose each element of the covariance matrix and and therefore have as much freedom as possible. In Figure 14 it is shown that the optimal choice for the number of gaussian components in this case would be 1.

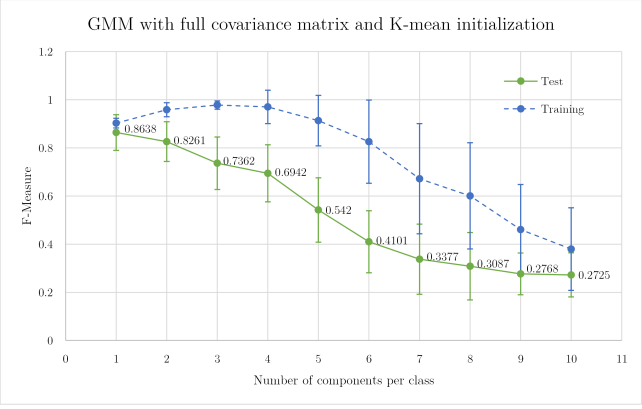
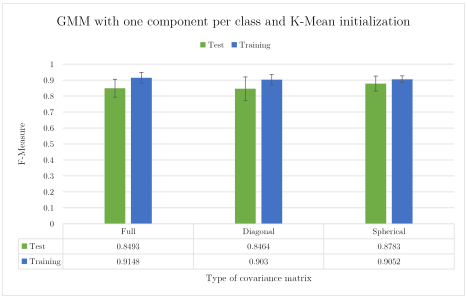
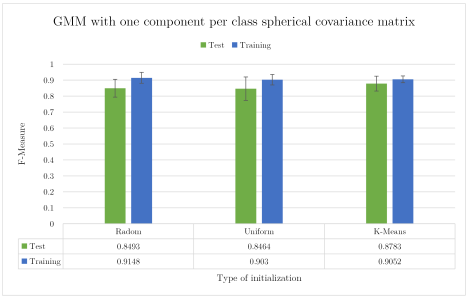


Figure 14: GMM: comparison for different number of gaussian components per class

The training set shows better results with a higher number of GMMs per class but the F-Measure for the testing set gets worse and the variance becomes higher, which is possibly due to an overfitting phenomenon. It can be noticed that for 4 and 5 components, the F-Measure's variance limit is higher than 1. As it is not possible to have a F-Measure higher than 1, this can be explained by the fact that the graph shows the variance values for a Gaussian distribution, while the data's distribution is not perfectly gaussian.



(a) GMM: comparison for different type of matrix

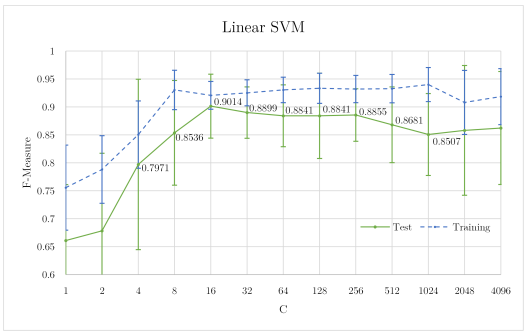


(b) GMM: comparison for different initialization

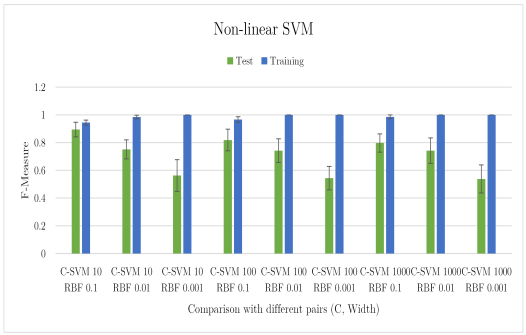
Figure 15: F-measure comparison for different type of matrix and initialization

By following the procedure explained in the block diagram in Figure 13, the plots in Figure 15 are obtained. In particular, they show that a spherical covariance matrix and a K-Means initialization method achieve the smallest variance on the training set. Indeed, a large variance on the testing set indicates an overfitting problem of the training set.

5.3 C-SVM classification (linear & RBS)



(a) Linear C-SVM: comparison for different penalty factor C



(b) C-SVM with RBF kernel: comparison for different pairs of hyperparameters

Figure 16: F-measure comparison for different type of matrix and initialization

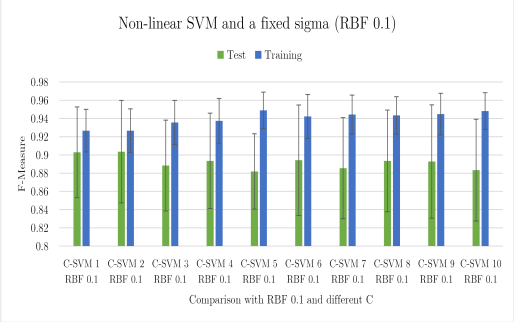
The hyperparameter that must be choosen for the SVM classification method are:

- the penalty term C;
- the width of the RBS kernel (RBS only).

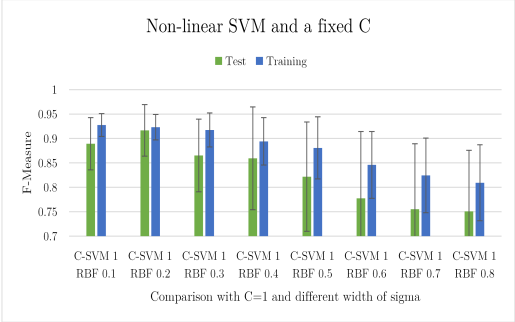
For the linear SVM, the C parameter was trained and tested for values going from 1 to 4096, 32 by 32. In Figure 16a it can be observed that the highest F-measure average for the testing set corresponds to $C = 16$, but in order to have a lower testing variance a $C = 32$ was chosen. Lower variance is preferable to minimize the overfitting of the training set even though the average of the F-Measure is slightly smaller.

For the non-linear SVM classification, values of σ and C are chosen as specified in Figure 13 and all combinations are compared in Figure 16b, which shows that the combination that optimizes the testing F-Measure is $C = 10$ and $\sigma = 0.1$ with the highest F-measure and the lowest variance. Indeed, it can be noticed that, for each chosen C, the testing F-Measure degrades and its variance increases as the σ decreases , while the training F-Measure increases and its variance decreases. This could be due to overfitting.

In order to verify what happens for lower values of C, the σ was fixed to 0.01 and values from 1 to 10 were tested (see Figure 17a). It can be observed that the higher the C, the more the training F-Measure increases and the testing F-Measure tend to decrease. This reinforces the hypothesis that



(a) C-SVM with RBF kernel: comparison for different penalty factor C and $\sigma = 0.1$



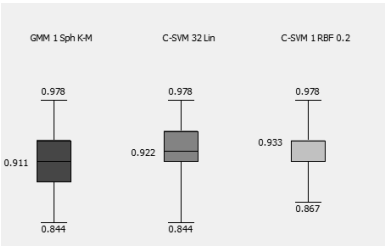
(b) C-SVM with RBF kernel: comparison for different width σ and a penalty factor $C = 1$

Figure 17: F-measure comparison for different type of matrix and initialization

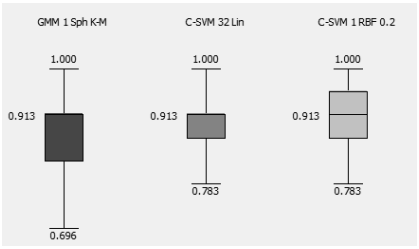
After testing several peer of hyperparameter, it is interesting to fix values of σ or C as in Figure 17. When σ is fixed, Figure 17a shows that the testing set is better when the penalty factor is relatively low. The F-Measure of the training set is slightly lower for small C and the F-Measure of the testing set is higher with a smaller variance. This result is explained because the dataset 2 is noisy and the factor C penalizes the misclassified points. Then, with a large C the model can overfitter the training set. Figure 17b shows very good result for $C = 1$ and $\sigma = 0.2$. However, it should be remembered that the SVM with a kernel is non-linear, which means that several pairs of C and σ can give an optimum. In this quantitative study the best pair found was $C = 1$ and $\sigma = 0.2$ but other optimal solution might exist.

5.4 C-SVM and GMM comparison

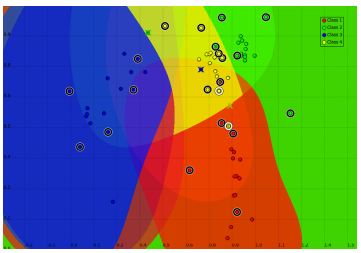
The results obtained for the three models gives good results and the SVM method with an RBF kernel is the best for classifying dataset 2. The variance of the testing set is lower and the average of the F-Measure is highest on the testing set for the SVM method with an RBF kernel (18b). Figure 18c shows the result of the non-linear SVM classification for the best value of C and σ founded.



(a) Training set



(b) Test set



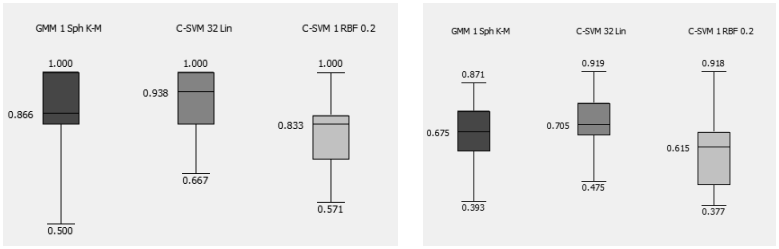
(c) C-SVM ($\sigma = 0.2$ and $C = 1$)

Figure 18: F-measure box-plot comparison with a train/test ratio of 66% and 30 folds for cross validation.

Once these results obtained, as indicated in Figure 13, it is interesting to test each model with different train/test ratio and number of fold for the crossvalidation.

Train/test ratio

As the dataset in question is quite small, the training/testing ratio must be carefully chosen. Indeed, the lower the ratio, the less the classifier will be trained and the more it will be prone to underfitting, which would cause poor performance on unseen data. This is a good approach for a dataset with enough datapoints. On the contrary, the higher the ratio, the more the classifier will be possibly overfitting to the training set and the variance for unseen data will be higher. A good trade-off was found to be the ratio 66%. Figure 19 shows that the classifier overfits the training set with a ratio of 10%.

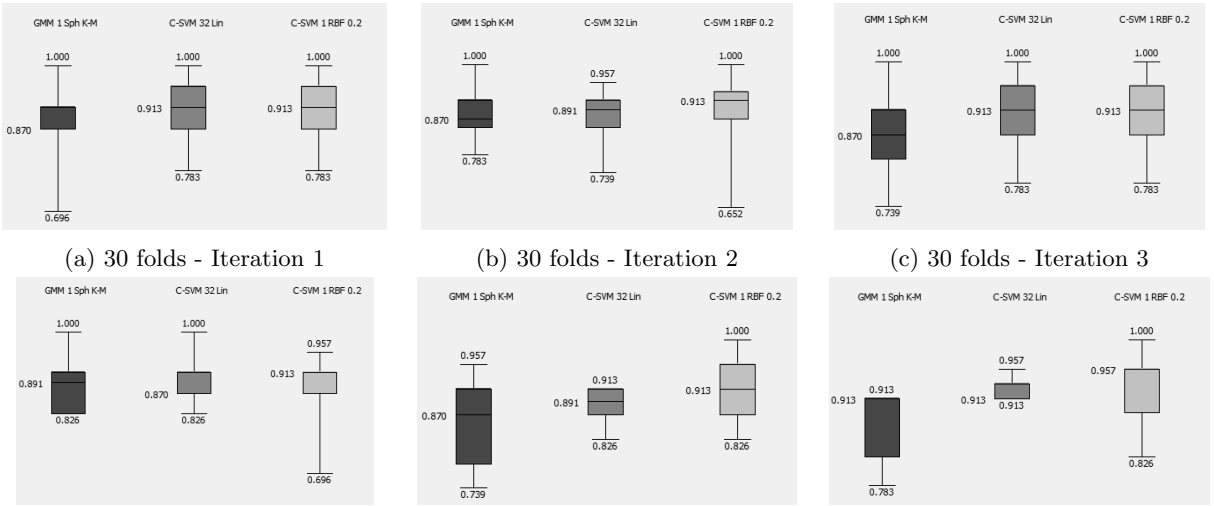


(a) Training set (b) Testing set

Figure 19: F-measure box-plot comparison with a train/test ratio of 10% and 30 folds for cross validation.

Number of folds sensitivity

In MLDemos, the crossvalidation method is based on a random separation of the data between training and testing sets on which the classifier is applied. The separation is then repeated f times and the results in the end are averaged. With this type of crossvalidation, a sufficiently large number of folds is recommended to obtain consistent results with lower variance. However, a too high number of repetitions is not useful because after a while the results do not improve but the cost of calculation increases. A good trade-off for this dataset was found to be 30 folds, as it was at this point that the results started to stabilize between several cross validation process. Figure 20 shows the disparity of the results between three iteration when choosing only 10 folds for the crossvalidation (Figures 20d to 20f). The results are more stable when choosing 30 folds as seen in Figures 20a to 20c.



(a) 30 folds - Iteration 1 (b) 30 folds - Iteration 2 (c) 30 folds - Iteration 3

(d) 10 folds - Iteration 1 (e) 10 folds - Iteration 2 (f) 10 folds - Iteration 3

Figure 20: F-measure box-plot comparison for 30 respectively 10 folds for crossvalidation and three different iteration with a train/test ratio of 66%.

5.5 Overall discussion and conclusion