Applied Machine Learning
Report on Practice Assignment

Team 40: Philipp Spiess, Lucas Waelti

Fall Semester 2017

1 Introduction

This report presents insights on different algorithms commonly used in Machine Learning for clustering and classification. Those are all based on a common dataset composed of pictures (cf. next section).

We will only use two classes in this report[1]. We made sure to provide a sufficient amount of samples per class (25 samples per class). As we will see later on, both classes are not easy to distinguish. This will allow us to observe limitations the algorithms that will be implemented in this report can present.

Note that sometimes, the dataset is flipped between Figures. The orientation depended on which computer the PCA was performed. We made therefore sure that the same orientation is maintained while performing comparisons within the different sections of the report.

2 Dataset

The dataset, as shown below in Figure 1, contains 2 classes of images:

- Motorcycles, 25 pictures
- Bicycles, 25 pictures

The bicycles as well as the motorcycles present the same orientation so that features inherent to the object are learned and not ones related to the orientation for instance. Some of the samples present some difference in orientation though, which can make classes less easy to distinguish. Some electric bikes tend to resemble the motorcycles while some motorcycles might be interpreted as bicycles.

3 Dimensionality Reduction

We then applied PCA to our dataset. We display here the first two generated eigenvectors in figure 2:

It is interesting to observe that the first eigenvector clearly identified the large and thin wheels of the bicycles while the second discarded this feature and considered the smaller and thicker wheels of the motorcycles. On the other hand, $e2$ gave more weight to the body of the motorcycles since bicycles are basically empty in this area.

[1]Dr. Anthanasios Polydoros confirmed the chosen dataset is adequate in the scope of this report.

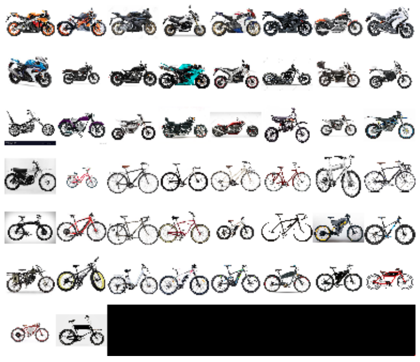


Figure 1: Dataset containing 2 classes of 25 samples each. The first class contains motorcycles while the second contains bicycles.

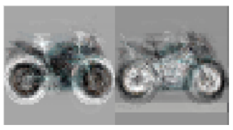
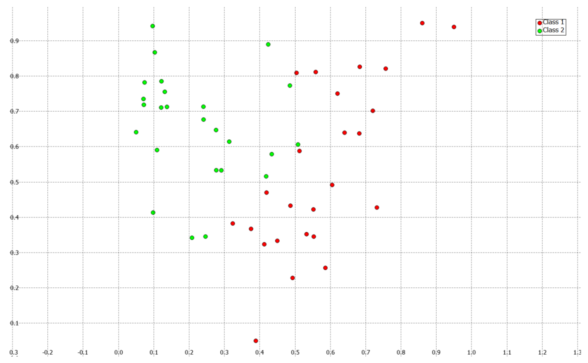
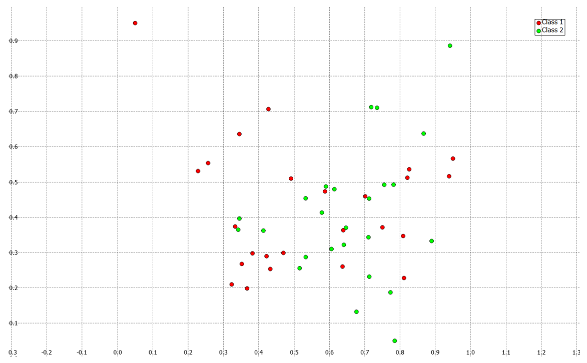


Figure 2: The first two eigenvectors $e1$ and $e2$ produced by the PCA

We can now observe which combinations of eigenvectors allow a classification of our images by observing the scatter plots of the projected dataset onto some pairs of eigenvectors. Shown in the figure 3 below, are two projection examples:



(a) A good projection of dataset (e1,e2). The two classes lie close to another but do not overlap.



(b) A bad projection of dataset (e2,e3). The two classes overlap and are indistinguishable.

Figure 3: Example of good and bad projections along different eigenvectors produced by the PCA.

What makes the subfigure 3a a good projection is the fact that the data is fully separated. There is no linear boundary between the two classes, but a curve can be drawn to create a boundary between the two classes. One could anyway define a linear boundary between the two classes, committing errors but providing a first approximation for the classification nonetheless. On the other hand, the subfigure 3b shows a very bad projection as the two classes are strongly overlapping. There are no ways of telling them apart since no boundary can be drawn between the two classes.

We will therefore base ourselves on the projection shown in Figure 3a for evaluating the different algorithms that will be discussed in the next sections.

4 Clustering

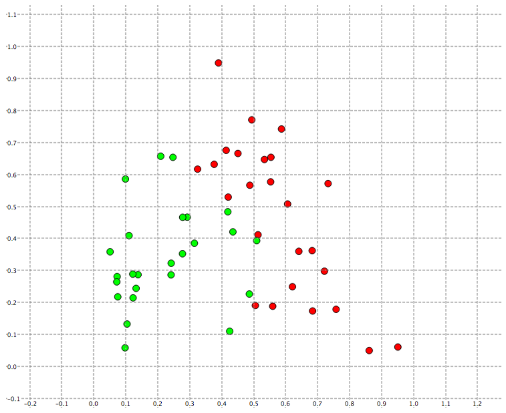


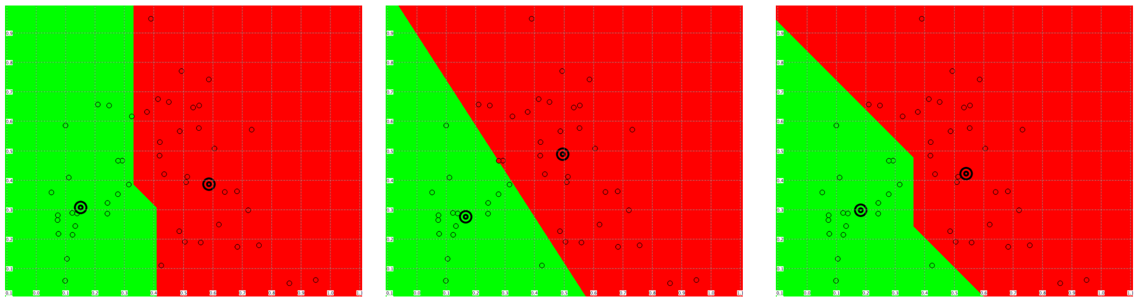
Figure 4: Data-set used to conduct clustering

4.1 Qualitative assessment of *Hard K-means*, *Soft K-means* and *DBSCAN*

4.1.1 Qualitative K-means assessment

With K-means, we can observe the effect on the algorithm of the *number of clusters K* and the *kind of norm* used (infinite, Manhattan, Euclidean or other...). The *sensitivity to random initialisation* is also an important property. Here we used the Euclidean norm, since it is well suited for 2 dimensional datasets, while testing the sensitivity to initialisation.

Effect of the norm The norm determines the way the distance from the centroids to the datapoints is calculated and thus will have an impact on the separation of the resulting clusters. While the L1-norm and Linf-norm generate segmented barriers, the L2-norm leads to straight lines separating the clusters. This is illustrated in Figure 5.



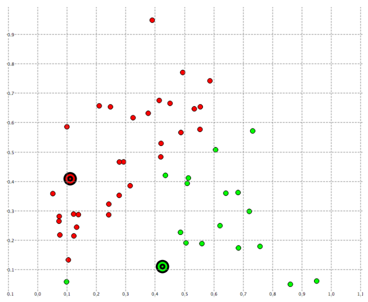
(a) *Hard K-means* using Manhattan norm (b) *Hard K-means* using Euclidean norm (c) *Hard K-means* using L^{inf}-Norm

Figure 5: Illustration of the properties of different norms after clustering with *hard K-means*.

As we can see when comparing the results of the sub-figures from Figure 5, the norm can drastically change the form of the boundary and therefore influence the performance of the algorithm. Basically, with different norms than the Euclidean norm, the clusters can have some more complex forms, which may allow them to better fit the shape of the data.

Effect of the number of clusters K In contrast with *soft K-means*, *hard K-means* cannot result in spots where multiple centroids overlap, there will then always be equally distributed clusters. So one has to know the number of classes to evaluate, in order to know how many clusters to use. With too many clusters, the data will be separated into smaller and smaller clusters and the clustering will no longer allow to generalise and extract common features across the data. *Hard K-means* would "**overfit**" in this case.

Sensitivity to initialisation Initialisation is very important for hard K-means. Because of a responsibility function taking only values of one or zero, centroids tend to not necessarily converge where they would be expected to. An example is provided in Figures 6 and 7.


(a) state after first iteration of k-mean

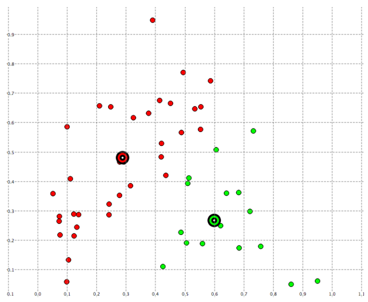
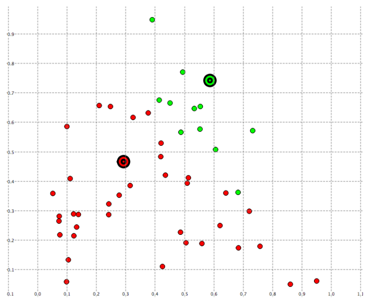

(b) state after convergence

Figure 6: Example where k-mean failed to cluster because of bad initialisation.


(a) state after first iteration of k-mean

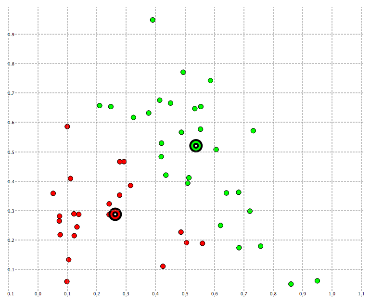

(b) state after convergence

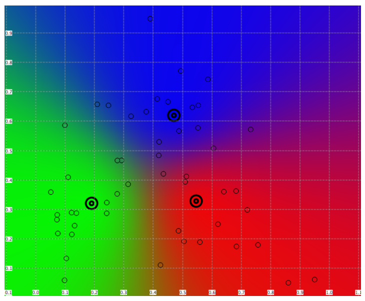
Figure 7: Example where k-mean clusters correctly after only 3 iterations, due to good initialisation.

4.1.2 Qualitative Soft K-means assessment

Here we can observe the effect of the *number of clusters K* and the *stiffness β*.

Stiffness For our dataset, with $K = 2$, the stiffness required to start distinguishing two clusters is approximately $\beta \approx 6.5$. For lower values the two centroids collapse onto each other and fail to produce two identifiable clusters. For higher values, for $\beta = 10$ typically, *Soft K-means* can produce better clustering than *Hard K-means* where the two classes are correctly separated. Indeed, *Soft K-means* can better deal with points that are difficult to assign. The initialisation of the centroids does not seem to have any significant effect here. Note that for high values of β ($\beta = 100$ for instance), the algorithm behaves like *Hard K-means*, since the datapoints are strongly attributed to a centroid or the other.

Number of clusters K We use here $\beta = 10$ to observe how K centroids can explain the data. With higher number of clusters, the data is clustered into at least three clusters. What we can observe in Figure 8 is that for $K = 3$, a general clustering pattern appears and no longer changes for higher values of K. This behaviour is very different from the one *hard K-means* displays with greater K.


(a) Clustering with *Soft K-means* with $K = 3$

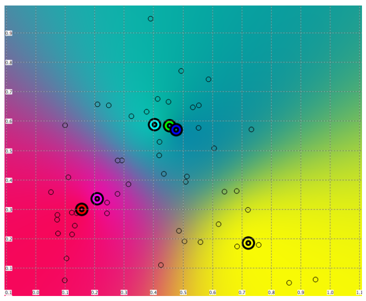

(b) Clustering with *Soft K-means* with $K = 6$

Figure 8: Clustering with *Soft K-means* with $K = 3$ and $K = 6$ ($\beta = 10$)

4.1.3 Qualitative DBSCAN assessment

Here we observe the effects of the ϵ (search radius around a data point) and $MinPoints$ (minimal number of points within ϵ) parameters. As *DBSCAN* is based on the distance between samples, our dataset is not well suited for a *DBSCAN* based clustering as the two classes lie very close to another.

ϵ parameter The distance ϵ allows to refine the clustering and have more complex shapes. It will also potentially increase the amount of clusters as well as the proportion of points considered as noise.

$MinPoints$ parameter If the minimal required number of neighbours within ϵ from a given data point is high, it will increase the chances of more isolated data points to be considered as noise instead of belonging to a cluster.

We can observe a result of DBSCAN for the parameters $MinPoints = 2$ and $\epsilon = 0.12$ on our dataset in Figure 9. Note that DBSCAN is not well suited in our case since the inter-class distance is smaller than the distance between datapoints in general. There is no way of actually properly cluster our dataset using this algorithm.

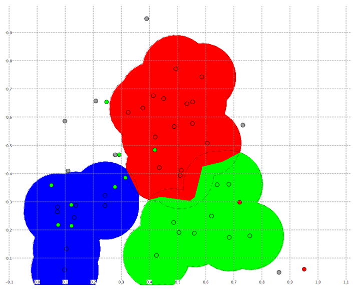


Figure 9: Clustering with DBSCAN with $MinPoints = 2$ and $\epsilon = 0.12$

4.2 Quantitative assessment of ***Hard K-means***, ***Soft K-means*** and ***DBSCAN***

We will use the AIC, BIC and F1 metrics to assess the algorithm here.

While AIC may have better predictive ability, BIC allows to find a computationally more efficient solution since the number of datapoints are included in the computation of the metric, which penalises the computational complexity. The AIC and BIC metrics are computed as follows:

$$\begin{aligned} AIC_{RSS} &= RSS + B, \quad B = K \cdot N \\ BIC_{RSS} &= RSS + \ln(M) \cdot B \\ \text{with} \quad RSS &= \sum_{k=1}^{K} \sum_{x \in C_k} |x - \mu^k|^2 \end{aligned} \tag{1}$$

with $B = K \cdot N$ (K: # clusters, N: # dimensions), and M: # data points.

The objective is to find a K that minimises the metrics.

For the F1-measure, the goal is to maximise it as $F1 \in [0, 1]$, 1 indicating a perfect clustering, where all labelled data-points of the same class are assigned to the same cluster. Furthermore, F1 in ML-Demos only uses a weighting of $\beta = 1$. As we saw in the course, there is a tradeoff between clustering correctly all datapoints of the same class into the same cluster and making sure that each cluster contains points of only one class. The F1-measure will allow to reflect this.

$$\begin{aligned} F_1(C, K) &= \sum_{c_i \in C} \frac{|c_i|}{M} \underset{k}{max}\{F_1(c_i, k)\} \\ F_1(c_i, k) &= \frac{2 \cdot R(c_i, k) \cdot P(c_i, k)}{R(c_i, k) + P(c_i, k)} = \frac{(\beta^2 + 1) \cdot R \cdot P}{\beta^2 \cdot R + P}, \quad \beta = 1 \\ R(c_i, k) &= \frac{N_{ik}}{N_{c_i}} \\ P(c_i, k) &= \frac{N_{ik}}{N_k} \end{aligned} \tag{2}$$

Note, C is the set of classes $\{c_i\}$.

4.2.1 ***Hard K-means***

We evaluate the recommended number of clusters K that the metrics recommend for different norms in table 1:

The norm does not influence the number of clusters recommended by the metrics but we see that AIC recommends one more cluster than BIC.

The fact of including labels indicates that only two clusters are desired. With a sufficient proportion of labels, the F1-measure indicates the correct number of clusters. With very few labels, it becomes more and more easy to achieve good results according to the F1-measure since only a few points are labelled and are therefore easily grouped within the same cluster, while belonging to the same class.

4.2.2 ***Soft K-means***

We proceed as we did for *Hard K-means*, this time evaluating the number of clusters K for different values for the stiffness β as shown in table 2:

Norm	AIC	BIC	F1-100%	F1-50%	F1-10%
L^∞	4	3	2 (0.83)	2 (0.8)	2 (0.84)
L^1	4	3	2 (0.8)	2 (0.79)	2 (0.85)
L^2	4	3	2 (0.8)	2 (0.82)	1 (0.89)
L^5	4	3	2 (0.83)	2 (0.85)	2 (0.89)
L^{10}	4	3	2 (0.79)	2 (0.87)	2 (0.89)

Table 1: Metric results for *Hard K-means* indicating the optimal K parameter. The F1 measures is given for 100%, 50% and 10% of training labelled data. The result of the F1 measure indicates the recommended number of cluster along with the value of the metric between parenthesis.

Norm	AIC	BIC	F1-100%	F1-50%	F1-10%
$\beta = 1$	5	2	1 (0.67)	1 (0.67)	1 (0.67)
$\beta = 6.5$	3	3	1 (0.67)	1 (0.67)	1 (0.67)
$\beta = 10$	4	3	2 (0.81)	2 (0.83)	2 (0.84)
$\beta = 50$	4	3	2 (0.82)	2 (0.9)	2 (0.92)
$\beta = 100$	5	3	2 (0.82)	2 (0.85)	2 (0.93)

Table 2: Metric results for *Soft K-means* indicating the optimal K parameter. The F1 measures is given for 100%, 50% and 10% of training labelled data. The result of the F1 measure indicates the recommended number of cluster along with the value of the metric between parenthesis.

4.2.3 ***DBSCAN***

Note: Nothing works on ML-Demos to evaluate this algorithm. It will therefore not be further discussed here.

4.2.4 Interpretation

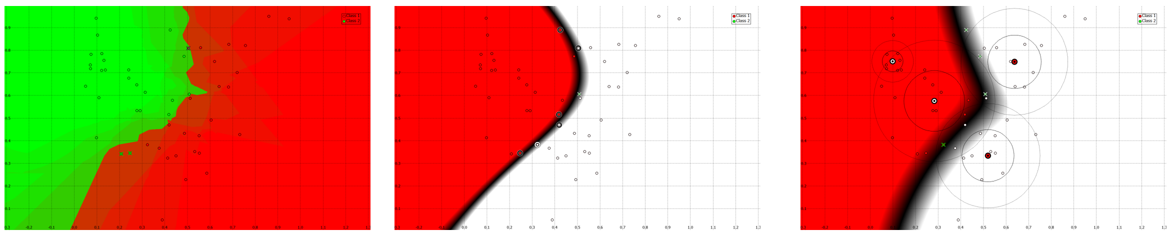
The fact that AIC recommends in general larger number of clusters corroborates the fact that AIC delivers better predictive ability while being computationally less efficient. Regarding *Soft K-means*, the stiffness value β tends to influence the number of recommended clusters. The bigger the stiffness, the bigger the number of clusters gets because datapoints are more strongly bound to a given cluster and the clusters therefore need to be smaller to better fit the data.

A tendency, regarding the F1 metric, is that the less labels are provided, the better the value of the metric gets for the optimum number of clusters. Indeed, the labelled datapoints of the same class have more chance of getting assigned to a same cluster since they are less numerous.

5 Classification

We will discuss here the hyper-parameters available for each algorithm and their respective effects. As a general remark on the training/testing ratio, a value of 66% should be adequate for our data-set of 50 samples. This yields a separation of 33 training samples and 17 testing samples. Higher ratios would not produce reliable enough results as the number of testing samples becomes too low. Choosing a smaller ratio is not adequate either as we risk to loose valuable information at the boundary between the two classes if not enough datapoints are provided for training.

We usually choose 30 folds during cross-validation to stabilise the result of the F-measure over this relative small set.



(a) k-NN with $k = 7$ neighbours (b) C-SVM with $C = 1000$ and $\sigma = 0.5$ (c) GMM with 2 spherical Gaussian

Figure 10: Results for the different classification algorithms

5.1 k-NN

k-NN actually offers 2 hyper-parameters but we will only evaluate the *first one*:

- Number of neighbours k
- Type of norm (euclidean here)

If k is chosen too high, it will produce under fitting. The algorithm will fail to identify local patterns and will generalise over the whole data-set. Figure 11 shows the performance of the algorithm for different values of k:

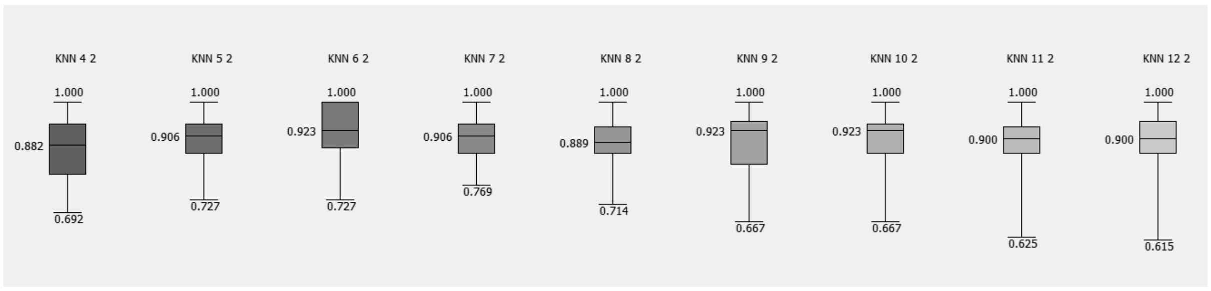


Figure 11: Performances of k-NN through 30 cross-validation, with parameter k ranging from 4 to 12 with Euclidean norm. Selecting $k = 7$ tends to be the best option here.

After comparing the results of the F-measure for different values of k, typically ranging from 1 to 15, for a training/testing ratio of 66% and 30 cross-validation, we observe that the best classification tend to be given for $k = 7$. The result can be observed in Figure 10a. Choosing a too high value for k risks to make the classifier underfit the model since it will be considering points too far from the currently evaluated data and will fail to reproduce the local behavior of the dataset.

5.2 C-SVM

C-SVM offers two hyper-parameters:

- C implements soft-margin. It will determine the cost associated to miss-classified points. If $C \to \infty$, this becomes hard margin SVM again.

- σ is the standard deviation of the RBF (Gaussian) Kernel and will define how sharp our decision boundaries must be.

The best F-measure that could be achieved on average for the training/testing ratio of 66% was with hyper-parameters $C = 1000$ and $\sigma = 0.5$ as illustrated in Figure 10b. In our case, we want a clear distinction between the two classes as they lie very close to another and penalise quite strongly miss-classification as it would induce more error because of the proximity of the two classes.

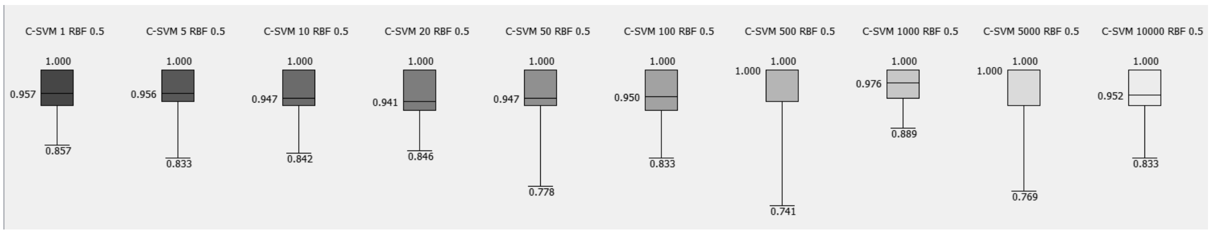


Figure 12: Evaluation of C-SVM through 30 cross-validations with parameter $\sigma = 0.5$ and varying $C \in [1, 10'000]$. We see that the choice of parameters $\sigma = 0.5$ and $C = 1000$ is adequate.

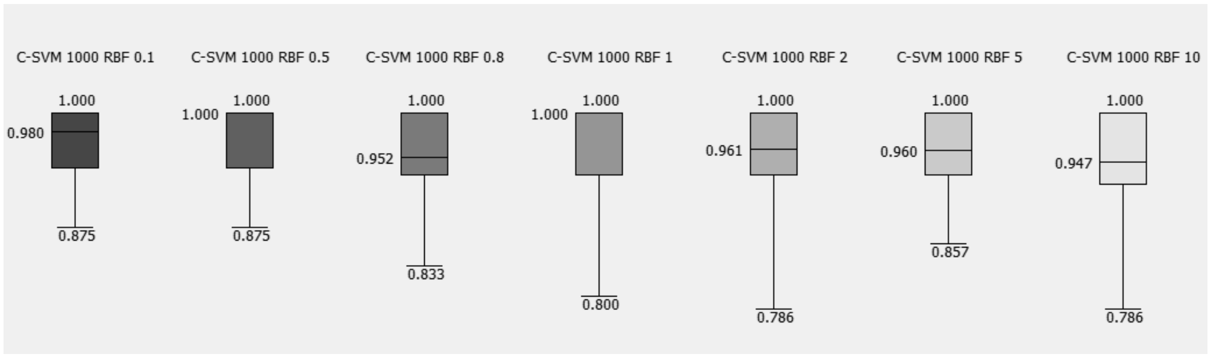


Figure 13: Evaluation of C-SVM through 30 cross-validations with parameter $C = 1000$ and varying $\sigma \in [0.1, 10]$. We see again that the choice of parameters $\sigma = 0.5$ and $C = 1000$ is adequate.

5.3 GMM

GMM offers two hyperparameters:

- N the number of gaussians per class.
- The type of covariance matrix: full, diagonal or spherical.

We used a training/testing ratio of 66% and a cross validation folds number of 10, as it seemed stable.

The results showed that the optimal number of gaussians per class was either 1 when using diagonal or full co-variance matrices or could be 1 or 2 when using spherical co-variance matrices as illustrated in Figure 10c. The full co-variance matrix did not result in better classification. The Figures 14, 15 and 16 confirm this.

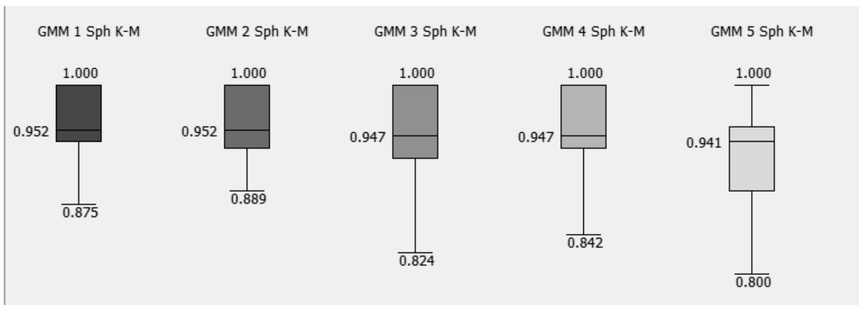


Figure 14: Evaluation of GMM through 30 cross-validations with spherical co-variance matrices. Selecting $k = 1$ or $k = 2$ provides in both cases good results.

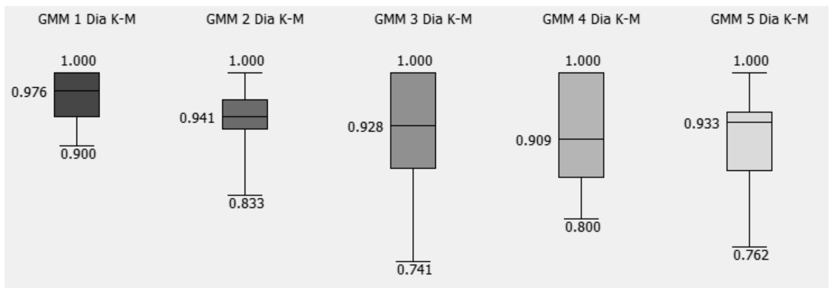


Figure 15: Evaluation of GMM through 30 cross-validations with diagonal co-variance matrices. Selecting $k = 1$ clearly yields the best results.

5.4 Performance comparison

We compared all methods by taking 30 cross-validation folds and it appeared that GMM produces better results than k-NN or SVM. It is expected in a sense where the algorithm is based on the properties of the data (its mean and co-variance). It should then be more capable of predicting the label of new samples. Furthermore, the two classes are balanced which is important for good results when using GMMs. However GMM requires more computational cost, so in an application with a lot of data, one will have to make a compromise between computational cost and precision and might choose SVM or k-NN instead of GMM. Although k-NN is exclusively memory based and does not learn from the training set, which has to be stored in order to produce a classification.

k-NN maybe tends to have more trouble because of the proximity of the two classes. The same goes with C-SVM that has to find a narrow and complex boundary, which makes it more difficult to systematically produce good results. The results are shown in the Figure 17 below, where GMM with 1 spherical Gaussian tends to outperform the rest of the algorithms:

6 Overall discussion and conclusion

Through this report, we had the occasion of experiencing with PCA, in order to extract the main features from a dataset, clustering, hence gathering similarities among the data in an unsupervised manner, and finally classification, where labels are (partially) provided.

The hyper-parameters of each methods have a great impact on the results and a search for optimality is always required to produce results as significant and useful as possible.

In a more general way, no matter if the goal is to cluster or classify the data, there are always different methods and approaches, which all present advantages and drawbacks. It will depend

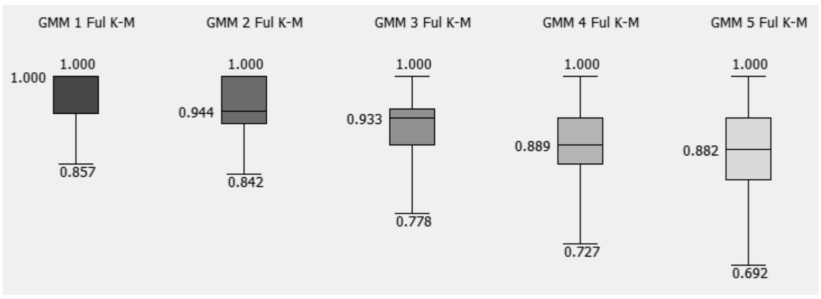


Figure 16: Evaluation of GMM through 30 cross-validations with full co-variance matrices. Selecting $k = 1$ clearly yields the best results.

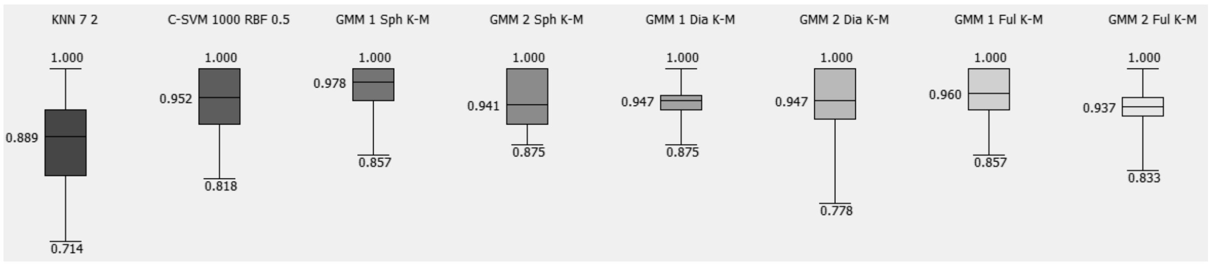


Figure 17: Comparison between k-NN, C-SVM and 6 configurations of GMM. GMMs outperform k-NN while the C-SVM algorithm produces sometimes as good results as certain choices of parameter for the GMMs.

on the type of data we are working on and it is therefore important to always consider which approach will bring the most insight on the information we want to extract from the data.