Project 3

Jean-Pierre Bianchi - GT ID jbianchi3

[ ]

1 Reminder of data sets

I am going to stick to my P1 datasets because they were specifically chosen to be able to test various algorithms, for the following reasons. The first dataset (DS1) is the 'Wisconsin Cancer' dataset [1] is a binary dataset with 699 instances and 9 features of breast cells with corresponding benign/malignant tumors labels. It is slightly imbalanced, 2/3 benign, 1/3 malignant. I selected this dataset because it gives 90+% accuracy right from the start with any algorithm, which means that, in theory, it should be not too complicated to compare how efficient feature reduction methods are, even if it's by only a few %.

The second dataset (DS2) is the 'White wine quality' dataset [2]. It has 4,900 instances and 11 features, and 10 classes, and highly imbalanced, so, for the sake of these exercises, I have split the classes to make it binary too. As I said in P1, the features represent a quite abstract (subjective?) criteria, ie the 'quality' of a wine. The predictors are all of chemical nature (acidity, sulphur, alcohol content, sugar etc), and I am not certain they are enough to get a perfect fit using any method since important information is ignored which for sure contributes to the perceived quality of a wine (tannins). This dataset is a bit 'risky' to say the least, which is why I picked an 'easy' first dataset, to be able to evaluate the algorithms on a more solid ground, and give a better feeling when analyzing the results on the second dataset in the right light. So all my analyses will be done on both sets at the same time.

From P1, I had the feeling that DS1 was going to be easy to reduce, and DS2 not so much, if at all, so I couldn't resist to take a peep at both datasets using TSNE. Fig 1 shows the result of TSNE on both datasets, and clearly the first one is going to be easy. For the second one, I tried various values of perplexity and early, small and big, I always ended up with two superposing clusters... Let's see what happens... Btw, I know that TSNE is a feature reduction algorithm, but here I've used it 'quickly' to have a feel at the data sets. I'm writing the report as I go so I am not sure yet what I'll take for the last feature reduction algorithm.

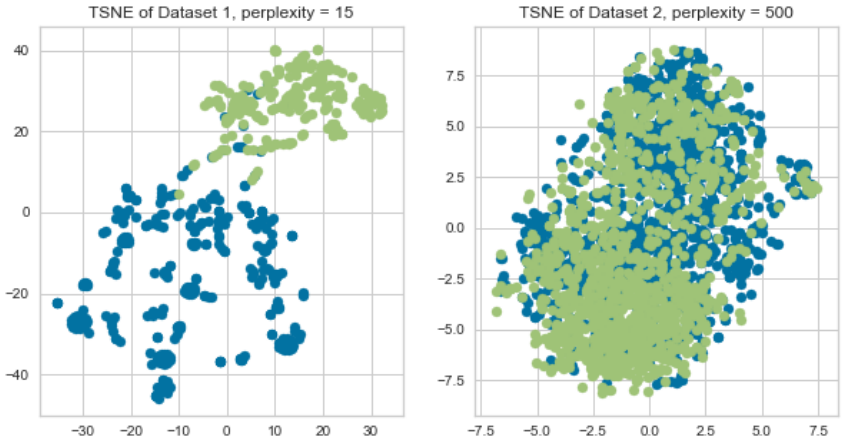

Figure 1: First glance at the datasets

In P1, we saw that every single time, the scorer that was automatically picked by the code for giving the

worst error, was F1, so I'm going to stick with it here too. Table 1 reminds us of the optimized Neural Network' F1 accuracy in P1.

	F1	Training time (ms)	Test time (ms)
DS1	96.5%	104.0	2.1
DS2	74.6%	281.1	2.0

Table 1: Neural Network accuracy without clustering or feature reduction from P1

2 K-Means and GMM

I have decided to not follow the order of the questions because I want to be able to analyze every algorithm one by one. So, first, I'll do 1 and 5 together, so I can get a first feel of K-Means and GMM on both my datasets, and directly apply a NN on the clusters labels. Then I'll do 2,3,4 on each algorithm, ie, first apply a feature reduction algorithm, then apply K-Means (KM from here on), GMM (for RM), and NN on it.

2.1 Clustering

First a few words about the metrics I have decided to use. I like the Silhouette metrics because it represents how much points are closer to their own cluster than to another. Well separated clusters should have a value close to 1. It's more about the geometry, because it doesn't say if the points in the cluster have similar labels (the ideal situation for classification). So I'll also calculate the homogeneity metric, which takes into account the ground truth (ie the target labels) to give the percentage of points in a cluster with the same label. The closer homogeneity gets to 1, the easier it gets to fit the data... I also calculated the AMI, because it accounts for chance and also the impact on Mutual Information when the nb of clusters is high.

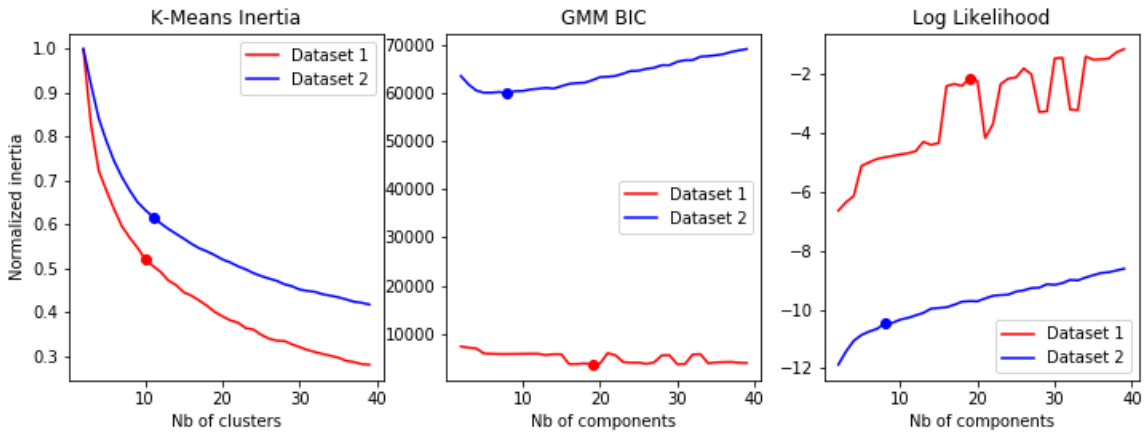

Figure 2: Inertia, BIC and Log Likelihood - elbow and optima indicated by big dots

Fig 2 shows the results of KM and GMM on both datasets. I have used the inertia provided by Scikit and the elbow method to determine the optimal nb of clusters, ie 10/19 (KM/GMM) for DS1, and 11/8 for DS2. For the elbow, I used code found on Github after evaluating the library yellowbrick, which gave me quite a few problems to install. I don't like much that method because it gives a different result if you give it 20 or 50 samples. But I didn't want to use the silhouette because it doesn't take into account the complexity, and with homogeneity, there is a risk to never reach a good value (as we will see below). In all cases, I will let the code decide the optimum values for k.

The high BIC value of DS2 probably comes from the 'k log(n)' part in the formula, ie it grows with the nb of samples, and DS2 is much bigger. Its log likelihood is not rising much though. I thought it was a bug, so I

switched the DS in my code (which processes them both at the same time), and I got the same results. I tried all the covariance types, but I always got the same clusters as shown in Fig 3 which give some hope for DS2. KM has managed to find some small clusters, which hopefully is noise or unimportant data. We'll know more when we reduce the dimensions. GMM is even better since it has created only a few prominent clusters, so maybe it has managed to make some sense of DS2 thanks to its non-spherical clustering capability, and we'll see that, in fact, the Neural Net fitting gives quite good results.

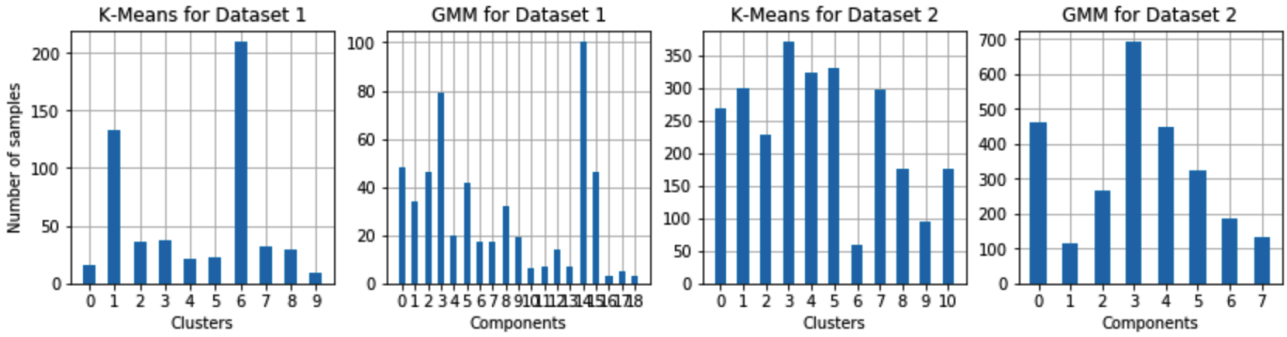

Figure 3: Clusters on raw data

For DS1, we can clearly see a good clusterization, with only a few predominant clusters. I was surprised by the number of clusters picked by GMM which could mean that GMM has done a better job and creates more clusters, many of which are small, so those could be useless data that will, hopefully be removed when we do feature reduction. Big clusters don't necessarily guarantee that all the points inside have the same label, but in this case, as you can see from Table 2, the homogeneity for KM and GMM are 0.8, so the clusters are pretty much in line with the labels, and we can see that the Neural Network managed to reach the same F1 score as in P1 (Table 2).

It is a stunning feat since I used only one feature, ie the cluster number. Because of that I had to do one-hot encoding, which not only creates many dimensions but also makes it hard for the solver to navigate through almost empty dimensions. It's the only explanation I've found to explain a fitting time 3 times bigger, although we started with one feature. Btw, Silhouette's formula has a bias towards round clusters, so I dropped it for GMM.

2.1.1 Neural Net fitting

	Dataset 1			Dataset2		
	P1	KM	GMM	P1	KM	GMM
F1	96.5%	96.4%	94.4%	74.6%	69.3%	67.8%
Time (ms)	104	335	393	285	1100	833
Silhouette		0.24	/	/	0.12	/
Homogeneity		0.83	0.85	/	0.13	0.09
AMI		0.43	0.34	/	0.06	0.06

Table 2: Metrics on KMeans and GMM clusters for both datasets

For the NN part, I used the labels, so I had to overload the 'transform' method of KMeans (and create one for GMM). Since the input values are so different than in P1, I had to optimize the NN every time. But it works quite well, especially for DS1. From Table 2, we can see that we didn't reach the same score as in P1 for DS2. I was hoping that GMM's superior ability to capture (non round) clusters would manage to do some magic with this difficult DS, but it actually did a bit worse than KM. It is still amazing to get a result like this with one feature only. Especially when we see a big drop in silhouette, homogeneity and AMI compared to DS1.

I'm seriously impressed. These three numbers being low at the same time confirm what we saw with TSNE (which was just a 2D projection), ie overlapping clusters, carrying information about both labels. I'm starting to get a better feeling as to why the algorithms from P1 had such a hard time with DS2, and I'm glad I kept it.

Btw, for the fun, I tried with only 2 and 4 clusters, and I still got an F1 of 93.8% and 96.4%, so, as we had already seen with TSNE, the data seems easy to 'split', and we should see a sizeable feature reduction. For DS2, I expect to drop only a few features at best since, as the homogeneity tells us, all the clusters seem to carry information from both labels. I forgot to mention that I am using a test set of only 20% of the data because I want to give as much data as possible to the NN.

2.2 PCA

PCA components are the easiest thing to calculate with Sklearn. Fig 4 shows the eigenvalues of both datasets. We can see how much faster they drop for DS1, to the point that I'll be trying to use much less than recommended by the code to see what happens. We can already see that we will need more principal components for DS2, which reminds us of how the clusters of DS2 were carrying information on both labels, making the classification more difficult, therefore less efficient.

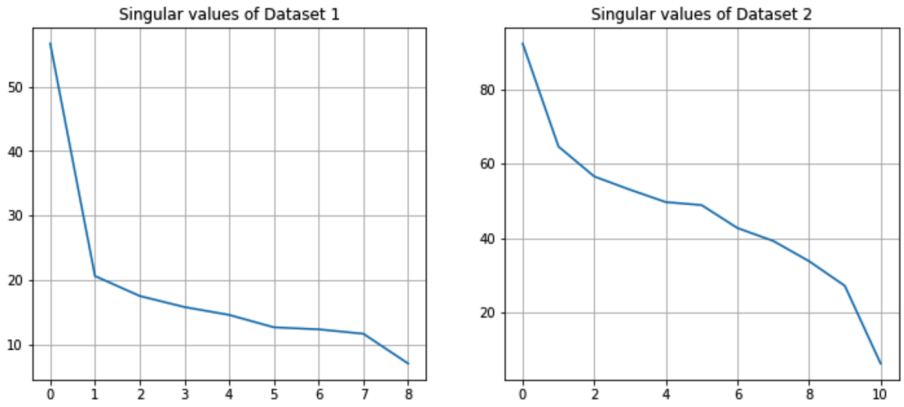

Figure 4: Eigenvalues of both sets

I ran two simulations as shown in Fig 5. 'Nb comp' indicates how many principal components we keep. We can clearly see that, the more we take, the more we capture from the total variance, and it goes quite fast, ie with only a handful, we already reach 80+%. To have a better feel, I also calculated the MSE (error from the reconstructed data), and also the F1 score on the test set. I started using NN but since we have to optimize at every step, it takes too long for what we're doing. I decided to use SVM because this is just to get a feel of how adding one more component influence the end result. To be clear, I did not use F1 to pick the nb of components, but a threshold of 90% that the variance has to reach. I chose a high value to try to keep most of the variance. As we can see, the code recommends 6 components for DS1, and 8 for DS2. The F1 scores already tell us that NN should, again, get great results, and DS2 will not reach P1's level.

2.2.1 Neural Net fitting

First, let's run our NN on both datasets, with results in Table 3. Let's start with DS2, whose result was surprisingly good, if we compare to Table 2. We gained 5% both on KM and GMM. My NN optimization is a bit coarse, to save time, so maybe I could have done even better. Dropping 3 components and improving so much with such a dataset is impressive. It confirms what was hinted by the kernel distributions earlier, ie this data is spread over several dimensions, but not as many as the nb of features. As a reminder, kNN got the highest score in P1, 81% with DS2, and kNN is robust to noisy data, so maybe this means that PCA has removed dimensions that were bringing mostly noise. We'll see below that, compared to 2.1, we have reduced

```
# Choose components to capture 90% of variance
n_com_wis = comp_var_f1(ds=1, msg = 'PCA analysis', learner='SVM')
```

```
PCA analysis of Dataset 1
Nb comp:  1 - total variance: 65.6% - MSE: 34.446% - F1: 95.5% - CPU time:   0.018 s
Nb comp:  2 - total variance: 74.2% - MSE: 25.787% - F1: 95.5% - CPU time:   0.014 s
Nb comp:  3 - total variance: 80.5% - MSE: 19.536% - F1: 94.3% - CPU time:   0.011 s
Nb comp:  4 - total variance: 85.5% - MSE: 14.452% - F1: 96.6% - CPU time:   0.011 s
Nb comp:  5 - total variance: 89.9% - MSE: 10.125% - F1: 94.3% - CPU time:   0.011 s
Nb comp:  6 - total variance: 93.1% - MSE:  6.867% - F1: 95.3% - CPU time:   0.012 s
Nb comp:  7 - total variance: 96.2% - MSE:  3.767% - F1: 95.2% - CPU time:   0.014 s
Nb comp:  8 - total variance: 99.0% - MSE:  1.004% - F1: 96.5% - CPU time:   0.010 s
Nb of components for variance > 90.0% = 6
```

```
# Choose components to capture 90% of variance
n_com_wine = comp_var_f1(ds=2, msg = 'PCA analysis', learner='SVM')
```

```
PCA analysis of Dataset 2
Nb comp:  1 - total variance: 29.6% - MSE: 70.434% - F1: 42.5% - CPU time:   0.594 s
Nb comp:  2 - total variance: 44.1% - MSE: 55.907% - F1: 47.5% - CPU time:   0.502 s
Nb comp:  3 - total variance: 55.2% - MSE: 44.837% - F1: 60.7% - CPU time:   0.438 s
Nb comp:  4 - total variance: 64.9% - MSE: 35.097% - F1: 66.2% - CPU time:   1.382 s
Nb comp:  5 - total variance: 73.5% - MSE: 26.541% - F1: 70.1% - CPU time:   0.360 s
Nb comp:  6 - total variance: 81.7% - MSE: 18.306% - F1: 72.2% - CPU time:   0.394 s
Nb comp:  7 - total variance: 87.9% - MSE: 12.086% - F1: 73.1% - CPU time:   0.365 s
Nb comp:  8 - total variance: 93.4% - MSE:  6.566% - F1: 73.7% - CPU time:   0.366 s
Nb comp:  9 - total variance: 97.3% - MSE:  2.688% - F1: 75.4% - CPU time:   0.359 s
Nb comp: 10 - total variance: 99.9% - MSE:  0.132% - F1: 76.0% - CPU time:   0.349 s
Nb of components for variance > 90.0% = 8
```

Figure 5: PCA simulations - DS1 (up), DS2 (below)

the nb of clusters as well.

About DS1, we can see that we can easily match P2's performance for DS1, BUT in less time, and interestingly, the running time decreases from 2 to 4 to 6 components. I guess it's not too bad to have more degrees of freedom to progress towards a minimum when the CPU can handle it.

Also, with 4 components only, we reach 97.7% which is the highest score ever, all methods combined. The best was 'boosting' that got 97.6%. I've run it several times, so I wonder how far it could go with a finer optimization. To me, this means that I could have chosen a lower threshold than 90%, as a visual inspection of Fig 5 already seemed to indicate. It's quite informative to see how much variance we can 'abandon' and still get excellent results! To be noted, also, the fact that with one less components, the F1 score for DS2 jumped over P1's mark, which is surprising, but shows, imo, how this kind of selection method has partially arbitrary.

	Dataset 1				Dataset2		
	P1	2 comp	4 comp	6 comp	P1	7 comp	8 comp
F1 (NN)	96.5%	91.4%	97.7%	96.5%	74.6%	75.1%	72.1%
Time (ms)	104	97	80	66	285	406	370

Table 3: Neural net classification on transformed data

2.2.2 Clustering analysis

To save space, I am not showing the inertia and BIC curves like in 2.1 because they are really identical, except that, as you can see in Fig 6, DS1 has now 8/19 (KM/GMM) clusters compared to 10/19, and DS2 has now 9/6 which is a significant reduction from 11/8. Also I am not showing the Silhouette, homogeneity and AMI because they are quite identical as in 2.1, so the same comments apply. You'll find them in my Jupyter notebook.

For DS1, GMM keeps asking for 19 clusters, although half of them are quite small, but I think it's because the BIC curve is almost flat, so with a slightly different mathematical criteria, we could have much less clusters. In the real world, I would probably manually pick a lower number, or change BIC for another, say AIC, but

for a project like this, I prefer to leave it as is, and see what maybe happens with the next feature reduction methods. It is weird because GMM works well on DS2 and did a great job at significantly and meaningfully reducing the number of clusters. To be noted, I had to use a different covariance type ('tied') for DS1 to get the beautiful GMM profile in Fig 6. If I had the time, I'd run NN on it.

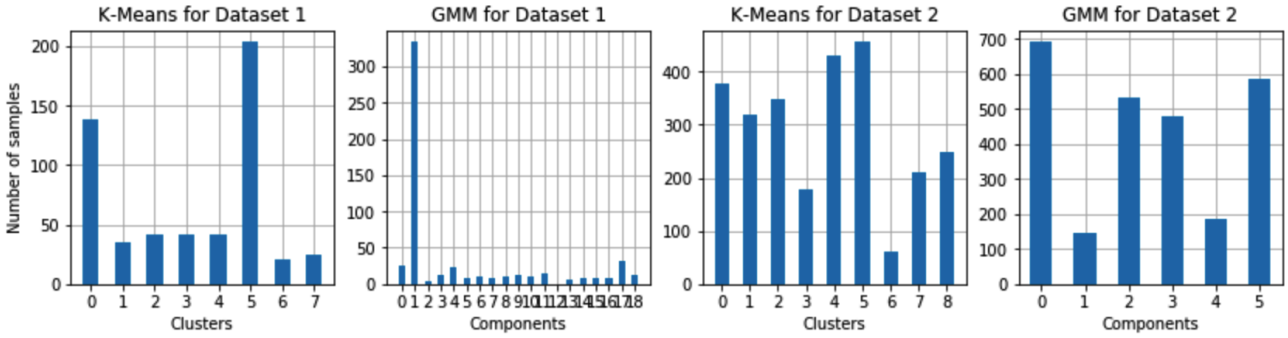

Figure 6: KM GMM clusters after PCA for Dataset 1

Since we had quite big changes with GMM, I looked at the clusters visually, to compare before and after PCA. With DS1, we see that the main cluster, on the left, is now much better isolated from the other clusters. This has to come from whatever PCA filtered-out which was 'attracting' other clusters, such as the small bue and red ones on the left. Also the whole layout looks much more 'well-divided' with smaller clusters sharing the most important data. We also understand why GMM requires 19 clusters, as it seems to target many isolated points in the top half and the right of the image. I didn't mention it earlier but the only metric that really 'jumped' was the silhouette, that went from 0.5 to 0.44. GMM or not, silhouette counts what it counts, and for sure, the change comes from the fact that the clusters surround their data better, especially the main one, and we don't have anymore so many situation where 3 clusters are very close to the same point.

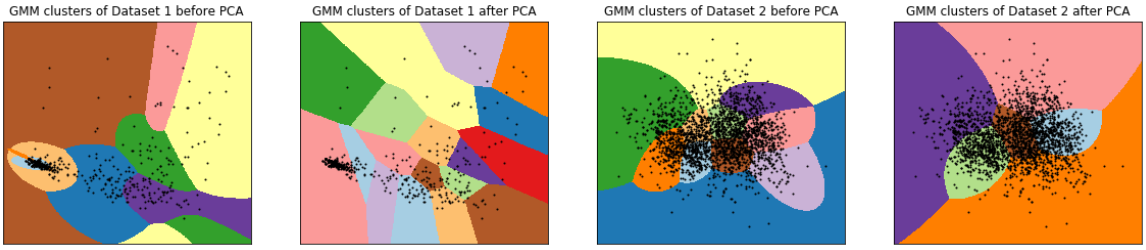

Figure 7: GMM clusters before and after PCA

We can't understand much from DS2's clusters, since we can see that all most of the points seem to make one big cluster. Although we can notice that GMM has managed to 'simplify' the landscape, eg by merging the brown and red cluster into the brown one, while improving the performance of the NN, again, for sure, because of the bad data that PCA left out. Let's move on to ICA.

2.3 ICA

2.3.1 Choice of k

I reviewed the possible choices for choosing k according to a kurtosis-based criteria, and the comments on Piazza. I tried to base my choice on the average kurtosis for each particular k. Some people recommend aiming for the maximum average kurtosis. I tried it and it worked beyond expectations. From Fig 8, we get an optimum k of 2 (TWO!!) and 5 for DS1 and DS2 respectively. The reconstruction rate of DS1 seems

particularly low ($\sim 25\%$), so let's see what happens when we feed the transformed data into a NN.

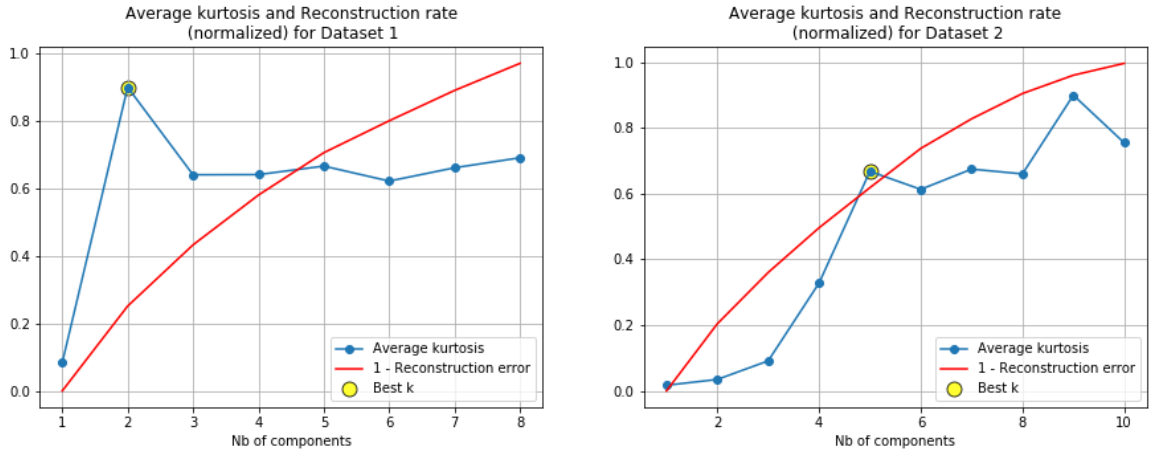

Figure 8: Average kurtosis and Reconstruction rate, normalized

2.3.2 Neural Net fitting

Table 4 shows the result of fitting the transformed data with a NN. The results are totally astonishing. With only 2 components, ICA managed to do just as well as PCA with 4 components (which was slightly better than the automatically selected value, 6). With 2 components, PCA was at 91% 'only'. And for DS2, with only 5 components, we almost match PCA with 8 components... The tendency of getting faster times for DS1 and slower ones for DS2 continues.

Since ICA is designed to separate things, I can only conclude that it has gotten rid of a lot of useless (noisy) features. After all, how could we explain the fact that we can reach the best F1 scores when a reconstruction loss of 75% for DS1, and 40% for DS2? The learning times are comparable to PCA's.

	Dataset 1		Dataset2	
	P1	2 comp	P1	5 comp
F1 (NN)	96.5%	97.4%	74.6%	70.6%
Time (ms)	104	82	285	559

Table 4: Neural net classification on transformed data

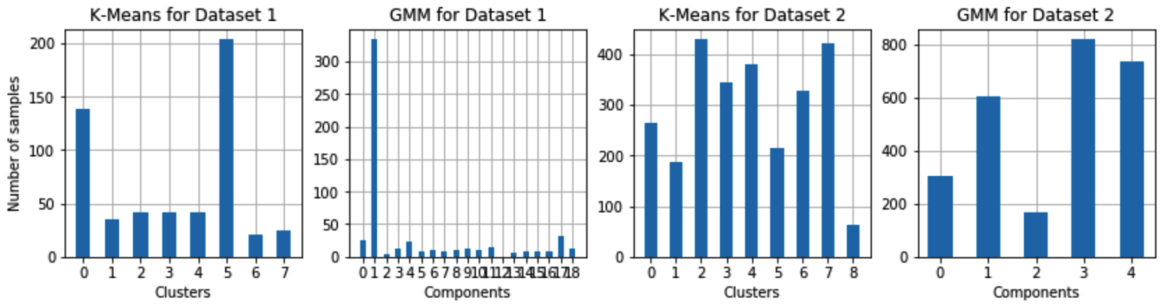

Figure 9: Clustering analysis

2.3.3 Clustering analysis

Let's take a look at what those clusters look like, in Fig 9. For DS1, we get something very very similar to what we got for PCA (Fig 6), except we have one less cluster for GMM on DS2, and one of them is very small. It really feels like DS1 has reached a plateau in terms of clustering, but DS2 keeps improving. The metrics are exactly the same as for PCA. There isn't much more to say here. I looked at the 2D projections but, there too, there isn't anything very different than we've seen already.

2.4 RANDOMIZED PROJECTIONS

2.4.1 Choice of k

This is a tricky one for me because my datasets have only ∼10 features. I decided to plot the reconstruction vs k, being done 10 times over each k to account for the randomness of this method. We can see that it is not an easy task to pick a number of components since the reconstruction error for any value varies a lot and can be quite low. However, we must succeed only once, so to avoid going for a high value of k to be on the safe side, I'll pick k = 4 for DS1, and k=5 for DS2, on the grounds that we've seen this type of method succeed even when the reconstruction was below 50%. We don't want all the data back with the noise and useless features anyway. I think it's the strength of this algorithm, to be able to shoot as many axes in all directions, to try to 'hit' groups of data randomly. Of course, it's not very attractive when you can use only a handful of components.

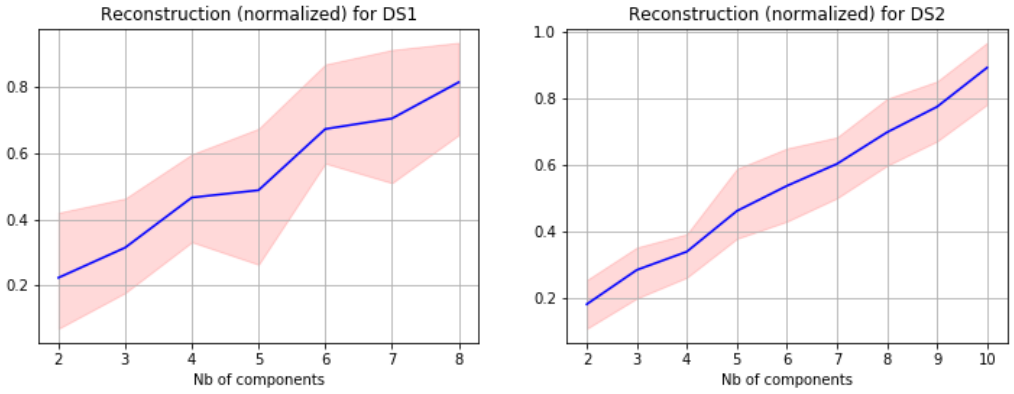

Figure 10: Reconstruction with max and min values

2.4.2 Neural Net fitting

By lack of time, this time, I will fit only DS1. I ran the NN 10 times, and the best score was 95.1% in 108 s for the fitting time (but I should probably multiply it by 10 to compare with another method). However, that was without optimization. With optimization, it reached 96.1%!! I tried with only two components, and the best I could reach was 91.6%, but maybe if I tried 100 times... So, again, I am not sure this method is suited for datasets with only a dozen features. However, since we are not limited here by the number of eigenvectors like PCA for instance, I couldn't resist to try with 50 or 100 components, and in that case, with one attempt, with optimization, DS2 reached 76% which is impressive.

2.4.3 Clustering analysis

I'm not sure what a cluster analysis on such a method can mean, but let's take a look at what those clusters look like, in Fig 9. During the NN optimization rounds, I was logging the seeds used, so I can at least initialize the algorithm with a seed that makes it work.
This time, I've shown the Inertia and BIC curves, and again, they are very similar to the ones I omitted. Unfortunately, that means that we have, again, 18 ridiculously small clusters for GMM in DS1. For DS2, the KM profile, in Fig 12, is similar, but the GMM profile has one more cluster than ICA, so 2 more than PCA. The metrics too are very similar. I have triple checked my code, to make sure I was not doing the same thing

over and over, re-using always the same data and algorithm for instance, but no. We have seen DS2 evolve and improve anyway.

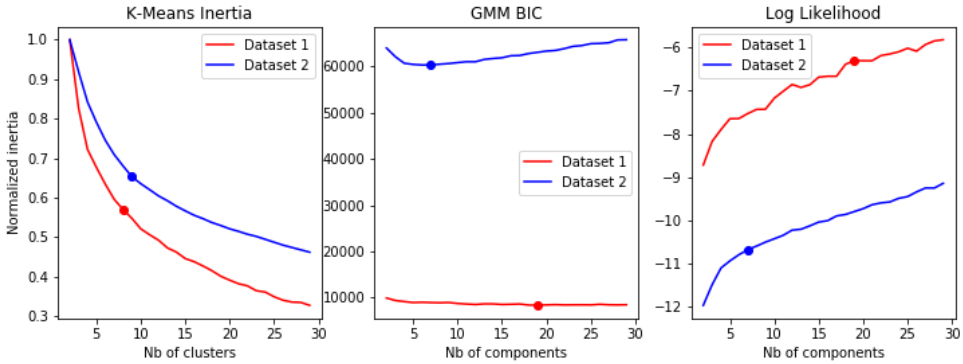

Figure 11: Inertia, BIC and Log Likelihood

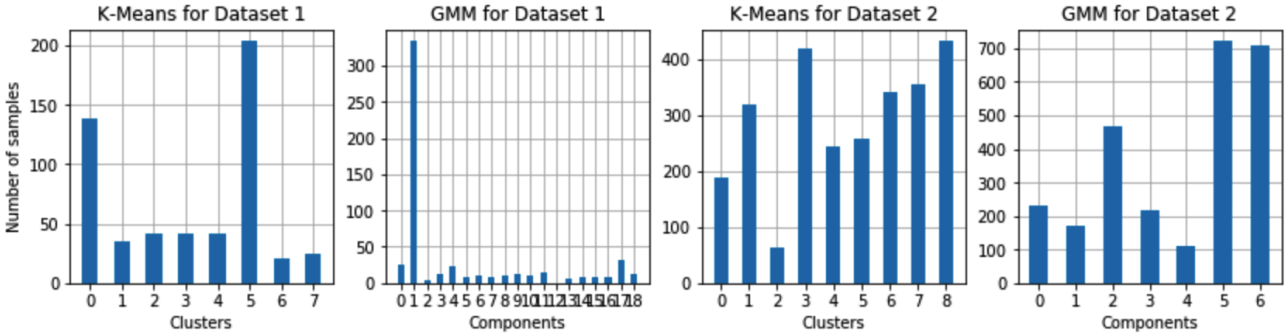

Figure 12: Clusters

2.5 KBEST

For my last algorithm, I've picked KBest because I was intrigued by how such a simple algorithm would compare to much more sophisticated ones, and it did quite well!

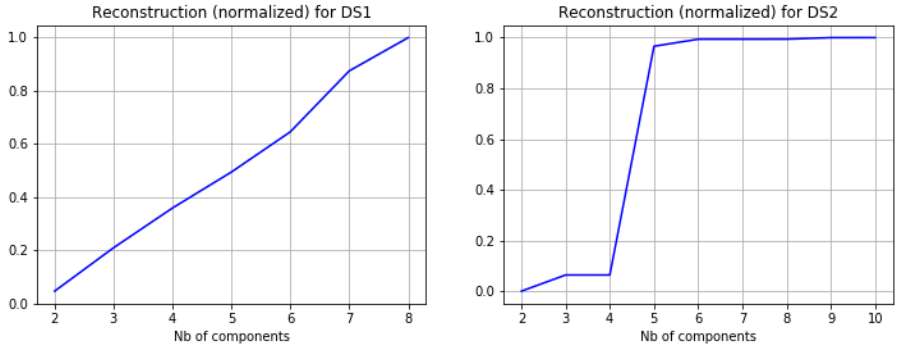

Figure 13: Reconstruction curves

2.5.1 Choice of k

To pick k, I used again the reconstruction curve because, after all I've done here, I ended up understanding that we don't need to be able to rebuild all the data very well, for the simple reason that we don't want the part of the data that was filtered out back. So, let's see what those curves look like, in Fig 13. For DS1, I'll pick k=3,

which allows to rebuild 20% of the data (k = 2 seems too extreme here), and for DS2, I'll take k=5 because, taking 4 seems a bit risky, and at 5, we have basically taken back a lot of the data but taking more doesn't bring anything else, so it will be noise for sure.

2.5.2 Neural Net fitting

In Table 5, we can see how well K-Best does for DS1, in almost half the time, which is normal since the NN works with much less features, therefore less weights to optimize. DS2 score is not so good, but here, the purpose was to evaluate a very simple feature reduction algorithm, and I think it did pretty well. Especially, taking into account how difficult DS2 is, with, I'm certain now, missing features as already explained (taste related features, such as tannins).

	Dataset 1		Dataset2	
	P1	3 comp	P1	4 comp
F1 (NN)	96.5%	90.1%	74.6%	64.6%
Time (ms)	104	66	285	355

Table 5: Neural net classification on transformed data

2.5.3 Clustering analysis

Here are the curves and the clusters. Once again the metrics are very similar and offer nothing to talk about. It feels like we've reached a limit as to what can be done to those two rather small datasets.

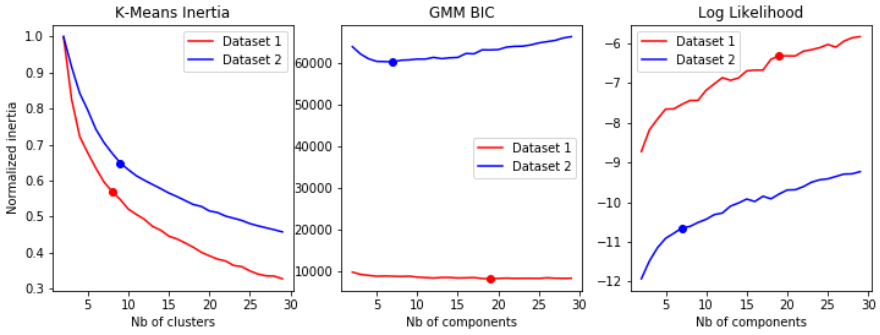

Figure 14: Inertia, BIC...

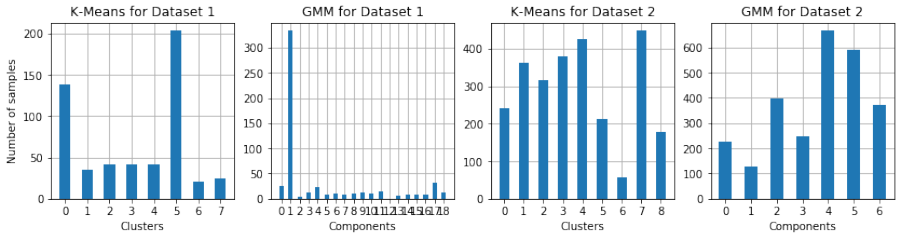

Figure 15: Clusters

3 Conclusion

Some results were really impressive, ICA in particular. I am very surprised as how the Inertia curves, BIC and metrics are so similar from one problem to the next, except at a few rare occasions. It feels like I reached a limit of the DS because no method beat the best results from P1. But I think most of those methods shine when we deal with hundreds of dimensions. My small datasets must have clobbered them in some way, especially RP.

4 References

[1] W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IST/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.

[2] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties In Decision Support Systems, Elsevier, 47(4):547-553, 2009.