# Unsupervised learning

BOISSIN Thibaut

## Table of contents

## I. Dataset selection

The first dataset chosen is the fashion MNIST, a dataset with 70k instances of 28*28 clothes pictures. Theses pictures are labelled with the clothe type. There are 9 different categories, ranging from shoes to shirts.This dataset is interesting for this problem, as it has numerous dimensions ( 784 dimensions ). Having an image based dataset, is also quite helpful for visualization purposes.
The second dataset is the WineEnthusiast dataset. This dataset contains 130k instance of wine review, including, text description, price, location. The instances are also given a score, which is our target. This dataset is interesting for it's text feature : in order to process it, a tf-idf vectorization has been used. The output of the description text processing is then a 130k * 31k sparse matrix, making it quite challenging (storing it as a full matrix would require 4g).
These data sets are quite complementary in their structure : one is dense and "small" (64M matrix by the way), while the other is very large but very sparse. Also as the data their represent is different, we expect those to highlight characteristics of the clustering and dimensionality reduction algorithms.

## II. Clustering

### 1. k-means

In order to choose the right amount of cluster, we must take a look at our data. As each cluster is determined by it's centroid, a centroid can be seen as an "typical individual" of the cluster. That's why, for fashion MNISTthe value of k should be the same order than the amount of classes, as the variance intra-class is not very high for most classes. On the Wine Enthusiast dataset, we need to think a bit further : as the distance used is the euclidean distance, and as the text is vectorized with a tf-idf, we can expect that text with same words in the same proportions to have a small distance. This will probably lead to clusters about the same kind of wine, as the words used to describe it will be from the same vocabulary.
To check this, two experiments have been driven : the first consist into performing a grid search on the k parameter and to measure the . The second consist into fitting a cluster and to visualize it's components.
The Fig 1 shows the score obtained for k-meansfor each value of k. The score is defined in sklearn as the default sklearn score, which is the SSE ( sum of normalized square differences between elements and their closest centroid). This measure has been chosen because it does not require the ground truth (by opposition to homogeneity and mutual information score). This is done so because we want to evaluate the impact of k, and having it different from the number of classes would make comparison with ground truth irrelevant.

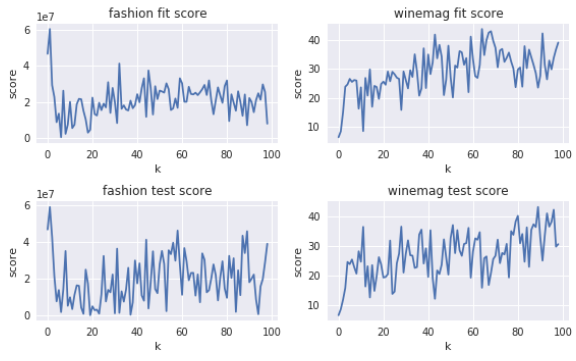


FIGURE 1 – influence of k on k-means clusters

It's interesting to note the presence of "noise" on the data, this may be due to the fact that some centroid may fall in between two clusters, depending on the choice of k. Hopefully the test scores follows the same tendency as the fit score (it only shows that we have enough data to have coherent clusters). On the fashion MNISTwe can see that the maximum ($k \approx 60$) doesn't not correspond with the number of classes (9 classes). This may be due to the fact that a class can be composed with several distinct elements (ex : flat heel and high heel boots). On the Wine Enthusiastdataset we can see that keeps increasing : text structure is more complex than pictures, as k increase, each cluster get a more specific set of words.

We can now visualize the cluster centroids by choosing a k values small enough to be printable, while being big enough to reflect the "standard individuals". For fashion MNIST, $k = 32$ has been chosen, the centroid are displayed in the Fig. 2. We can see that these images are weighted average of the class element. As the images are pretty neat, it shows that the clusters are consistent.

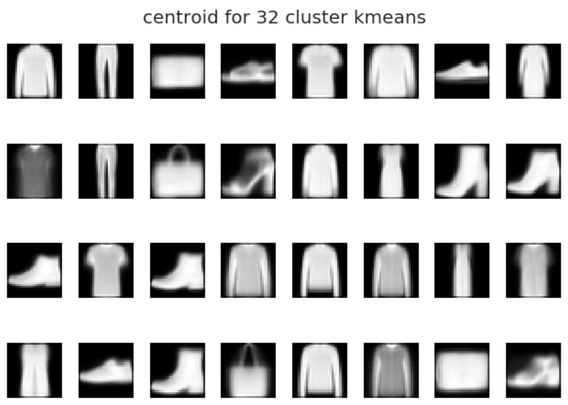


FIGURE 2 – k-means clusters visualization for 32 clusters

As the Wine Enthusiastdataset has a lot of columns, a high number of clusters is recommended to have only a few words per cluster (for display purposes). The Fig. 3 shows the bag of words associated to a random picked cluster.

['2018' 'aftertaste' 'and' 'associated' 'at' 'black' 'cherry' 'core' 'does' 'drink' 'from' 'fruit' 'fruits' 'fruity' 'full' 'have' 'is' 'it' 'lighter' 'noir' 'normally' 'of' 'pinot' 'red' 'rich' 'ripe' 'same' 'style' 'substantial' 'tannic' 'the' 'time' 'wine' 'with']

FIGURE 3 – k-means cluster example Wine Enthusiastdataset

We can note that the words can be present in the clusters, for 2 main reasons :

— the word is rare in the corpus
— the word has many occurrences in the texts of the cluster

It confirms that performing feature selection before performing the clustering would have been great, as some irrelevant words are present in the clusters. Also by looking at the variance of the number of words in the cluster (the standard deviation value is 1157 words !) we can see the major drawback of k-means : it assumes that all clusters are the same size, leading to have super specific cluster (many words in the cluster) aside some super vague clusters (few words clusters) this assumption is obviously not true on sparse data.

## 2. Expectation maximization

The Expectation maximizationalgorithm implemented by sklearn make the same strong assumption : that the distribution if a Gaussian mixture. For data coming from sensors this assumption make sense, but in problem such as computer vision or natural language analysis, this assumption is not trivial. To figure it out, a grid search has been processed. This experiment highlighted some point about EM : first, this algorithm is computationally expensive ( compared to k-means, which is used to initialize the distribution means ). Even 24h computation time on a 32 cores computer didn't succeed to handle this heavy dataset. Second, it's doesn't work with sparse data ; there is no sklearn implementation for this, and having one wouldn't be relevant as sparse data leads to an excessive number of features. This is why Expectation maximizationcouldn't be performed on the Wine Enthusiastdataset. To do so, an other pre-processing of the data would be required as tf-idf leads to sparse data. The use of the *BayesianGaussianMixture* would also give more appreciable results as it implements a Dirichlet distribution as a prior.
The Fig. 4 shows the evolution of the per-sample average log-likelihood according to the evolution of the number of components.

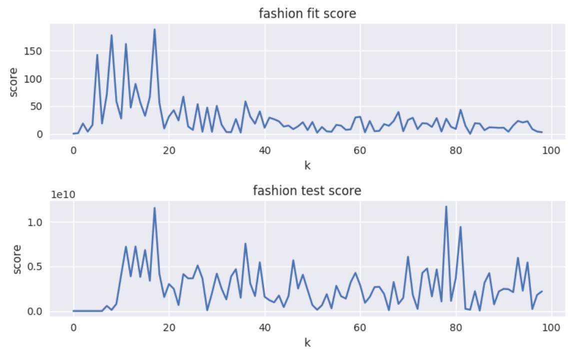


FIGURE 4 – influence of k on Expectation maximization clusters

As we can see the fashion MNISTtends to give good results for smaller amounts of features ($k \leq 20$ ) which shows that the Gaussian mixture model fit the data better than k-means would do. To check this we can take a look at some samples generated by the computed distribution, as done in the Fig. 5. We can then see that the distribution is able to compute new samples although there are still some confusion between classes leading to strange pictures.

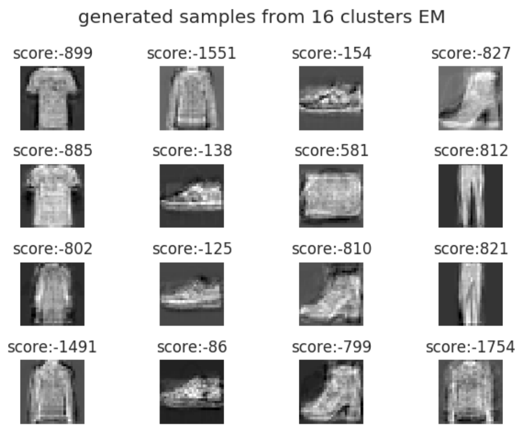


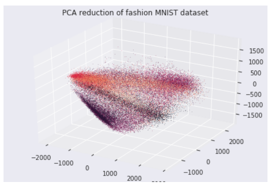
FIGURE 5 – kmeans clusters visualization for 32 clusters

## III. Dimension Reduction

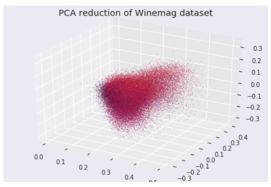
### 1. PCA

As PCA performs dimension reduction using SVD, it does the assumption that features with the most variance bring the most information to the system. This make the assumption that the features are independents and also assume the noise on the data to be Gaussian identically on all the features.

Firstly the Fig. 6 shows the 1000 highest singular values for both data sets. As we can see fashion MNISTis very dense as 90% of the singular values are greater than 100, on the other hand, the Wine Enthusiastdataset is, as expected, very sparse as the singular values decay shortly.



(a) 3D PCA reduction on fashion MNIST



(b) 3D PCA reduction on Wine Enthusiast

FIGURE 6 – 3D PCA reduction of data sets

In order to visualize the impact of the the reduction dimension, the space will bee reduced in 3d space (which is of course too low for other purposes ). The Fig. 7 shows the reduced space, where the color of each point correspond to it class. As we can see the points of the same class arrange themselves around lines. It confirms that the classes are mostly determined by the components which are maximizing the variance. The Fig. 8 shows the images associated to singular-vectors, interestingly the obtained images recall the frequency domain encoding used in jpeg. The main singular value reveals large features while vectors associated to smaller singular values shows thinner details. As expected each vector represent a mix of all classes.

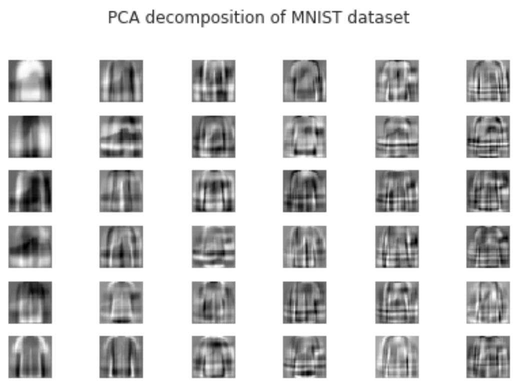


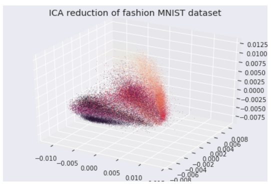
FIGURE 7 – 36 first PCA base vector for fashion MNIST

### 2. ICA

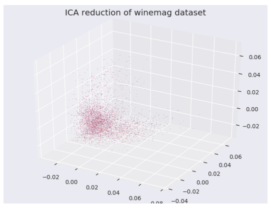
ICA assume the signal to be a linear combination of hidden variables (blind sources). This is why we expect the ICA to have the following behaviour :

— on fashion MNIST : find features that independently compose many clothes (heel, sleeves are examples)

— on Wine Enthusiast : find independents word sources characterizing wines ( smell vocabulary, appearance vocabulary are examples)



(a) 3D ICA reduction on fashion MNIST



(b) 3D ICA reduction on Wine Enthusiast

FIGURE 8 – 3D ICA reduction of datasets

The Fig. 10 shows that the components found on fashion MNISTare related to an element class : the image lead to nicely separated clusters. On the other hand we can see that the found sources used to describe a wine are not related to it's score ( which can be seen as the color of the point shows it's score ).

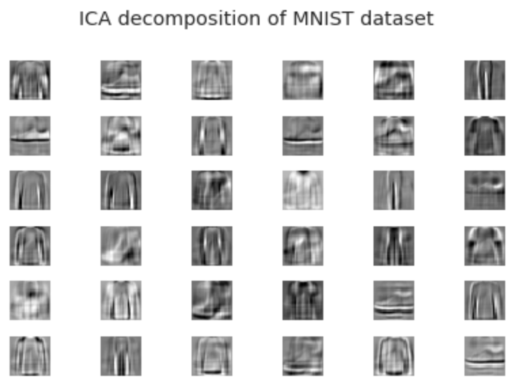


FIGURE 9 – 36 ICA base vector for fashion MNIST

As we can see on Fig. 8, ICA reveals independent features that compose an image : we can see sole on some images, sleeves or bag handles on others. This decomposition has also it limits : it assumes that each element is a linear combinations of the found sources. In practice, elements of each class is composed from a distinct set of ICA features, this is why fitting an ICA on each class, an combining all the results would certainly lead to a better decomposition.

**About tf-idf :** The code contains an ICA decomposition of the Wine Enthusiast dataset. This experiment revealed 2 points : Firstly, the sklearn ICA don't work on sparse matrix this is why performing an ICA on the full dataset became impossible. Secondly, the ICA decomposition on the 5000 first row of the dataset revealed that the tf-idf don't behave well on short texts as it become very sensitive to word repetitions. The bag of word visualization showed only numbers and years on most features. Also as tf-idf give floating numbers, many sources can use the same word while staying independent : leading to sources using all the words with different tf-idf proportions.

## 3. RP

Random projections have two main advantages : it is fast, and it is agnostic. Even if it is limited in the reduction factor, it can help on large data sets. One important result about RCA it the Johnson-Linderstrauss lemma which allow to compute a minor bound of the dimension to use in order to have a major bound of the measure distortion. This allows to keep the properties of the original space while working on reduced space.
The Fig. 10 shows sample of 3d reductions on the fashion MNISTdataset (the result on the Wine Enthusiastdata set is available in the $fig/$ folder, but the reduction is so drastic that the results don't add information this figure doesn't )

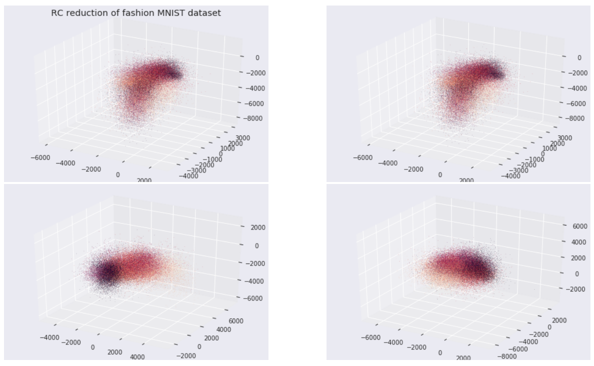


FIGURE 10 – 3D RCA reduction of fashion MNIST

The lower the dimension is the higher the variance of the result is : according to the Johnson-Linderstrauss lemma, the distance distortion is bounded by the reduced space dimension. This means that reduction in 3D is doesn't preserve well the distances between elements, explaining the very different shapes of the clusters. However even with this distortion the cluster formed by each class is still visible although they tend to mix.

# IV. Dimension reduction and clustering

The idea to combine dimension reduction and clustering can be relevant, on two axis :

— dimension reduction may improve quality of the clustering as it try (if the assumption made are relevant) to remove noise. It also would give better results on small dataset by avoiding bellman curse of dimensionality.
— dimension reduction can improve computation time, as the clustering algo-

rithm works on smaller data
The Fig. 11 shows the evolution of the results of k-means in terms of score and computation time accordingly to $k$. The metric chosen is the *normalized mutual information score* as we have the ground truth, and as we don't want the ordering of the labels to interfere with the results. The closer this score is to one, the closer this distribution is to the ground truth.

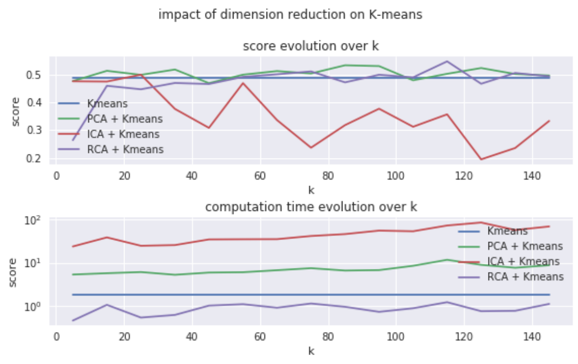


FIGURE 11 – results of K-means after dimension reduction

As we can see the PCA improves the results of k-means, this may be due to the fact that PCA tends to maximize inter-class variance while minimizing intra-class variance. k-means is apparently disturbed by some noise which is associated to small singular values.
Surprisingly the ICA seems to lower the score obtained. As ICA defines independence as non-Gaussianity, it would explain why k-means have lower results as it model only spherical clusters.
Also we can note that Random Projection reduction has great effects in terms of performance. The results become comparable to the bare k-mean, but this is at the cost of a variance on the results if the dimension is too low. In order to avoid it, we can use the Johnson-Linderstrauss lemma to compute the minimum dimension while keeping the distortion on the measure bounded.

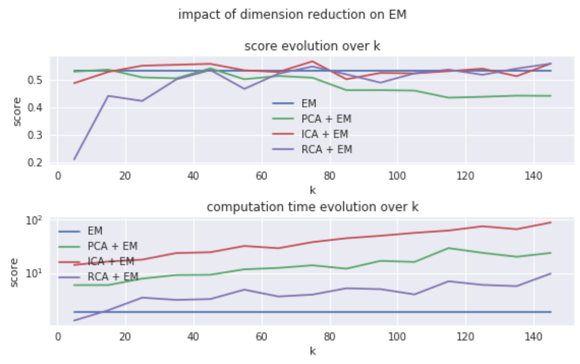


FIGURE 12 – results of EM after dimension reduction

The Fig. 12 shows the results of the same experiment with the EM algorithm, the main difference with k-means is in the modeling of the distribution. As the EM used here model a Gaussian mixture, we would expect the output of the reduction algorithm to be a assimilable to Gaussian mixture, in order to get good results.
We can note the PCA experience the same effect as the ICA did with the k-means algorithm. The more dimension you take into account the worse the results are with EM. This is because in this case the computed $E[z_{ij}$ is not accurate under Gaussian Assumptions. This would explain why the result starts to drop when $k \geq \#_{cluster}$
Also we can note that, in term of computation time, the gain made previously on random projection is quickly lost ($k \geq 10$). This is mainly because EM is less sensitive to the input dimension (the cost over dimensions is small compared to the cost over the number of instances).

## V. Dimension reduction for neural net

The idea is to take advantage of the dimensionality reduction to improve the results of a Neural Network. First we can use the Haussler theorem to check the dimension of the neural network :

$$m \geq \frac{1}{\epsilon}(4 \log \frac{2}{\delta} + 16(r+1)s \log(es) \log \frac{13}{\epsilon})$$

By setting $\delta = 0.01$, $\epsilon = 0.85$, with 3 layers of 50 perceptrons we have $s = 150$ and $r = 50$

and then :

$$m \geq 4.93\ 10^5$$

The Fig. 13 shows the performances of the same neural network applied to the output of the dimension reduction algorithms.

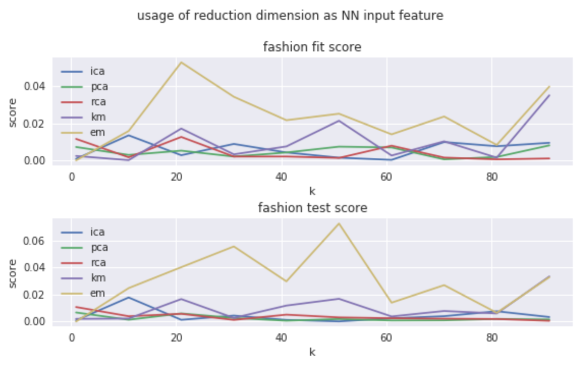


FIGURE 13 – results of NN after dimension reduction

For readability, the score of the bare neural network is not displayed in this graph as it is very negative ( an error in the Haussler theorem application ? ). This is certainly the case as the two algorithm giving the best results are the EM and k-means, which both output a single dimension ( the cluster column is the used alone as input of the neural net ). It reveals that the NN in not suitable for high dimension input space, having two larger layers would have certainly improved the results.

## VI. Conclusion

This project permitted to highlight the characteristics of these algorithms :

**K-means** algorithm has the following advantages :

— only a few parameters to tune
— has efficient implementations (MiniBatchesKMeans support stream data flow for instance)

But k-means also have the following drawbacks :

— produce ellipsis cluster
— rely on a normalization computed from training data, it assume there is enough data per feature to compute the variance.
— cluster have similar size, categories must have similar cardinality in the training set.

**Expectation maximization** has the following advantages :

— allow a better modeling of the distribution
— it can take into account relationships between variables ( using Bayesian net )

On the other hand,

— it's computationally expensive
— it doesn't work well by default on sparse matrices

**PCA** has the following properties :

— reduce the space while keeping components with high variance
— some noise associated to small singular values can be removed with PCA ( but this is not a general rule)
— the output of PCA tends to break the assumptions of the EM algorithm (Gaussian mixture)

**ICA** can decompose the data into independents sources :

— leads to good results if the data can be modeled with blind sources
— is computationally more expensive than PCA
— behave poorly on sparse matrices (sklearn don't even have implementation for it )

**RCA** is quite interesting on some aspects :

— it is very fast, useful to combine with algorithms where the complexity is linked

to the space dimension.
— results tends to be close to the full space with appropriate choice for k.

**Finally** clustering and dimension reduction can be seen as way to "condense" the data. this can be done for various purposes : data explanation (with the clusters/components visualization), unsupervised classification, and data preparation for other type of learning.

**This project** has been very interesting on other points such performance issues, dealing with sparse matrices wasn't easy but instructive. Also this showed limits of each algorithm, as always, there's no free lunch !