Representation Learning

Loïc Albarracin - Nicolas Rahmat

***Abstract*—**

How to allow a machine to think like a human ? A machine can only execute programs. Its brain is a long chain of transistors instead of neurons. As early as the 1950s, we are witnessing the development of the first Machine Learning (ML) algorithms. Nowadays, there are two types of ML algorithms : supervised and unsupervised algorithms. The first learns from observations. For example, if you show him a picture, it will be able to tell you if it corresponds or not to something he knows. The last clusters samples from real life, and humans don't have to inform it about the data. In a sense, it can be said that unsupervised ML algorithms learn on their own. The main problem is still the dimension of the space in which the initial dataset belongs to. This is called the "curse of dimensionality", which is well solved by using algorithms such as the PCA algorithm.

I. Introduction

The main goal of Machine Learning is to build intelligent machines, also called Artificial Intelligence (AI). We want that machines take "good decisions". In order to do that, they need knowledge. This knowledge, our humans' one for example, can neither be transmitted verbally nor write in a program. However, something exists, which is very useful to allow machines to get this knowledge. It is observations [1]. Moreover, Machine Learning doesn't mean learning by heart, but it means generalizing from examples, where probability mass concentrates. Here, the "guess-work" is to know which new structure or strategy makes sense. In fact, most of the time, data is contained in a high-dimensionality manifold. By fighting this dimensionality problem, machines allow to discover underlying causes and factors, to explain data. Naturally, there is another huge problem : machines will never learn enough examples to cover all data in the world (e.g. functions). This problem is known as the curse of dimensionality problem. To solve it, bins are defined with probability of appearance. For example, with images, on each point, it is possible to apply a geometrical transformation, like a rotation or a translation, with the aim to get new images. It means that a manifold could be created from all of these images. To improve Machine Learning, data has to be sorted. Here, human work is to prepare the input features, which are essential for successful Machine Learning, which represents about 90% of total effort. There is also an opposition between handcrafting features and learning features. The goal of representation learning is to guess the features, the factors and the causes to finally get a good representation. In regards to deep representation learning, algorithms attached to it attempt to learn multiple levels of representation of increasing complexity and abstraction. When the number of levels can be data-selected, this is called Deep Learning, using several neuronal layers of computation. But a main problem has to be raised, knowing that there are different parameters for each distinguishable region : how many particles must we have to define a cluster? And more generally : how can we make clusters without seeing the data? To illustrate this we will base our article upon scikit-learn [2], a Python library of machine learning.

II. Methods

In this part we will describe the different methods we use to harness the machine learning power for analyzing images and statistical models of the brain.

A. Decomposition methods

PCA : Principal Component Analysis

The main idea of the PCA algorithm is to compress a set of data from a high dimensionality space into a low dimensionality space [2] The transformation $\mathbf{T} = \mathbf{XW}$ maps a data vector x_i from an original space of p variables to a new space of N variables which are uncorrelated over the dataset. However, not all the principal components need to be kept. Keeping only the first p principal components, produced by using only the first p loading vectors, gives the truncated transformation

$\mathbf{T}_L = \mathbf{XW}_L$. $(L \ll N)$ To truncate we use the SVD (Singular Value Decomposition) algorithm :

$$\mathbf{X} = \mathbf{U\Sigma W}^*$$

with U is a unitary matrix from $\mathcal{M}_L(\mathbb{K})$, $\Sigma \in \mathcal{M}_{L,p}(\mathbb{K})$ has diagonal coefficients in $\mathbb{R}_+$ and other coefficients reduces to 0. W^* is the adjoint matrix of W, a unitary matrix from $\mathcal{M}_p(\mathbb{K})$. The $\sigma_i = \Sigma_{i,i}$ are the singular values.The columns of U and W are respectively the left and the right singular vectors of M. The squares of the singular values of M are the eigenvalues of MM^T and of M^TM. The initial problem can now be simplified by using these theorems : $\mathbf{T} = \mathbf{XW} = \mathbf{U\Sigma W^*W} = \mathbf{U\Sigma}$

The output obtained with the PCA is a set of successive orthogonal components that explain a maximum amount of variance. Other data will then be projected on L orthogonal components, learned by the PCA algorithm.

ICA : Independent Component Analysis

The aim of ICA is to separate a multivariate signal into additive subcomponents that are maximally independent. [2] Typically, ICA is used for separating superimposed signals, instead of being used for reducing dimensionality as we can see on Figure 1

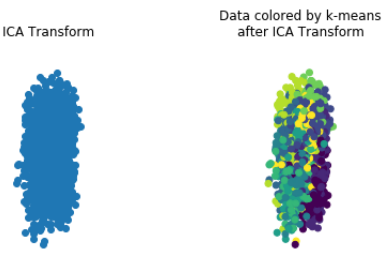

Figure 1: ICA Decomposition, KMEANS Clustering

B. Clusters methods

KMeans

When someone uses the KMeans algorithm, he tries to separate samples in a certain number of groups of equal variance. [2] Note n this number of samples. This is done by minimizing a criterion known as the *inertia* or *within-cluster sum-of-squares*. This is called the *clustering of the data*. To use this algorithm, the number of clusters at the arrival has to be precise. When the number of samples rises very high, this algorithm scales well. It has also been used in many different fields across a large range of application areas.

The KMeans algorithm takes a set X of N samples as input. This set will be split into disjoint clusters C. The μ_j value represents the mean of the samples which belong to the j^{th} cluster. The means are the so-called cluster "centroids". In general, the centroids are not points from X, although they belong to the same space. The main goal of the KMeans algorithm is to choose centroids that minimize the *inertia*, or *within-cluster sum-of-squares* criterion :

$$\sum_{i=0}^{n} \min_{\mu_j \in C} \|x_j - \mu_i\|^2$$

The KMeans algorithm is also called Lloyd's algorithm. basically, the algorithm is divided into three different steps. The first one chooses the initial centroids. To do this, we just have to choose k samples from X. After this, the algorithm loops between the two next steps. The second step assigns each sample to its nearest centroid. The last step creates new centroids by taking the mean value of all of the samples assigned to each previous centroid.The difference between the old and the new centroids are computed and the algorithm repeats these last two steps until this value is less than a threshold. In other words, it repeats until the centroids do not move significantly.

C. Metrics

Adjusted Rand index (ARI)

$$\overbrace{ARI}^{\text{Adjusted Index}} = \frac{\overbrace{\sum_{ij}\binom{n_{ij}}{2}}^{\text{Index}} - \overbrace{[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}^{\text{Expected Index}}}{\underbrace{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}]}_{\text{Max Index}} - \underbrace{[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}_{\text{Expected Index}}}$$

Given the knowledge of the ground truth class assignments and our clustering algorithm assignments of the same samples, the ARI is a function that measures the similarity of the two assignments [2], ignoring permutations and with chance normalization, perfect labeling is scored 1.0. Random (uniform) label assignments have a ARI score close to 0.0 for any number of clusters and any number of samples. No assumption is made on the cluster structure, it can be used to compare clustering algorithms such as KMeans which assumes isotropic blob shapes with results of spectral clustering algorithms which can find cluster with folded shapes.

Homogeneity Score

Given the knowledge of the ground truth class assignments of the samples, it is possible to define some intuitive metric using conditional entropy analysis. [2] Homogeneity (meaning that each cluster contains only members of a single class) and completeness (meaning that all members of a given class are assigned to the same cluster [2]) are two desirable objectives for any cluster assignment.

We define homogeneity h and completeness c as :

$$h = 1 - \frac{H(C|K)}{H(C)} \qquad c = 1 - \frac{H(K|C)}{H(K)}$$

where H(C|K) is the conditional entropy of the classes given the cluster assignments and is given by :

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \cdot \log\left(\frac{n_{c,k}}{n_k}\right)$$

and H(C) is the entropy of the classes and is given by :

$$H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{n} \cdot \log\left(\frac{n_c}{n}\right)$$

with n the tital number of samples, n_c and n_k the number of samples respectiveley belonging to class c and cluster k and finally $n_{c,k}$ the number of samples from class c assigned to cluster k. The conditional entropy of clusters given class H(K|C) and the entropy of clusters H(K) are defined in a symmetric manner.

D. Datasets

The Digits dataset

This dataset is made up of 1797 8x8 images. [2] Each image, like the one given below, is of a hand-written digit. In order to use an 8x8 figure like this, we would have to first transform it into a feature vector with length 64.

The functional Magnetic Resonance Imaging (fMRI) dataset

The fMRI dataset is composed of signals which represent statistical models of the brain response to neutral pictures and highly negative emotional pictures. [3]

The matrix contains 241 observations which are each 23880 voxels (volume elements from an MRI image). Each voxel is a cube with an edge of 4 mm. First we recreate the corresponding gray matter mask that was used to generate this data, resampled with the corresponding dimension : 4X4X4 mm.

III. Results

The Digits dataset

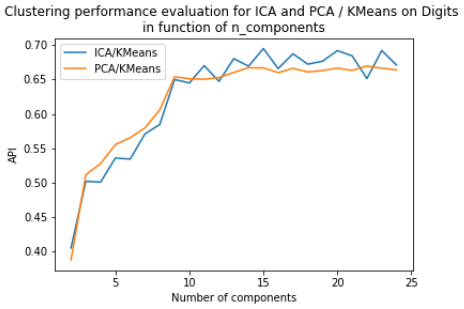

Figure 2: PCA and ICA Performance

Figure 2 represents evolution of ARI for ICA and PCA. After doing a PCA or an ICA on a dataset of digits and after trying to cluster with KMeans algorithm, the performance on Digits with PCA/KMeans is growing up to ARI = 0.667 and homogeneity score = 0.736, for n_components = 15. The clustering performance on Digits with ICA/KMeans is globally growing until n_components

= 15, where it reaches up to ARI = 0.667, and homogeneity score = 0.737 before beginning to diverge.

It appears that ICA and PCA are both efficient algorithms. Of course, they are not better than the supervised method, but they reach an ARI between 0.60 and 0.70 and a homogeneity score between 0.7 and 0.8 so these results, being close to 1, we conclude that the performance is pretty correct.

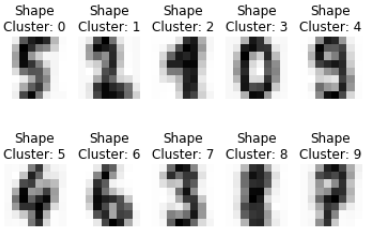

Figure 3: The corresponding representation of the KMeans centroids

As we can see in the Figure 3, there are 10 clusters and the centroid of each one got a distinguishable shape. Even if we did not sort the clusters from 0 to 9 in increasing or decreasing order, we can see that the unsupervised algorithm delimit clearly 10 different clusters. Hence the difference between unsupervised and supervised performance, which is evaluated by the precision (0.97) for the supervised and the API closeness to 1 (0.7) for the unsupervised, could be explained in the placement of certain sample in the correct cluster.

The functional Magnetic Resonance Imaging (fMRI) dataset

First of all to know if it is possible to clustering the data with the unsupervised we will test this set with a supervised one : SVM.SVC.

In a first approach, we will select a subset of the voxels by using the SelectKBest (function of the library : sklearn.feature_selection [2]) estimator [3]. It runs a univariate feature selection, by performing an Analysis of Variance [3] for each voxel. This will output the predictive value of each voxel. Therefore, we keep the k-voxels with the highest predictive value, a good starting point is to keep about 10% of the total features, so $k = 2000$.

Figure 4 depicts the coefficients of the SVM, which shows the importance of the voxels.

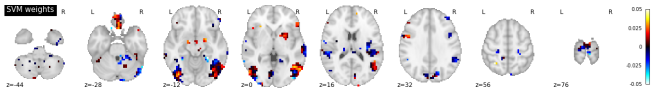

Figure 4: Results of supervised algorithms (SVM.SVC) on Digits

These regions correspond to the ones that are important to discriminate between neutral and highly negative pictures.

As we don't have a Training set and Testing set, and as there are more "1" ratings than "5" ratings, we will use the "permutation_test_score" function to estimate the accuracy. This function estimates the chance level by running classification on multiple (default= 100) random permutations of the labels in y.

The first output of "permutation_test_score" is the accuracy without permutation. The second output is a vector giving the accuracies on the permuted versions, and the last output is the probability that the actual classification is at chance (so the lower the better, typically we expect $p << 0.05$) So a simple SVM trained on the 2000 best voxels yields an accuracy of about 92%, which is significantly ($p < 0.01$) better than chance level (estimated between 64% to 72%).

The results obtained for the unsupervised algorithm are very close to the reality. Indeed, among the 241 labels given to the KMeans algorithm, only 28 received the wrong one, meaning that they were not put in the right cluster. The fail ratio of this experiment reaches 11.6%.

But by measuring the performances with the metrics previously presented, the result is worse than the one just presented. It is high possible that the problem encountered comes from unbalanced classes [4].

IV. Conclusion

Machine Learning algorithms are nowadays very beautiful computer tools, especially when it deals with the recognition of symbols like digits. The aim of our work was to extend our comprehension of those algorithms, more precisely of those which use unsupervised methods of learning. In this context, we learned to use pipelines with algorithms such as PCA and ICA, which transformed data before giving it to the KMeans algorithm, to gather them in cluster as would any unsupervised Machine Learning algorithm. These algorithms have this advantage over those supervised that they do not need any database for their learning. The disadvantage that follows is that the accuracy of the results given by the unsupervised algorithms is lower than the one of the supervised algorithms. The proposed unsupervised approach yielded ARI scores lower than those of supervised approach, which suggests a lower performance of the clustering. In particular, using the digits dataset, we obtain an ARI of 0.69 and a homogeneity score of 0.77 which is orders of magnitude higher than with the fMRI dataset. This difference could be due to the difference of the dimension of the space in which both datasets are living. Our draft of unsupervised algorithm analysis is not exhaustive: it consists only of a small part of the unsupervised algorithms and of these, we analyzed only a minimal fraction of their real capabilities. All in all, unsupervised algorithms offer a partial answer to a new type of human problem concerning more particularly the grouping, the sorting, the judicious choice of data on a structure. This structure can be decomposed into several layers. And about this complex structure we would talk about Deep Learning.

References

[1] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," *JMLR: Workshop and Conference Proceedings*, vol. 27, pp. 17–37, 2012.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] L. J. Chang, P. J. Gianaros, S. B. Manuck, A. Krishnan, and T. D. Wager, "A sensitive and specific neural signature for picture-induced negative affect," June 2015, main article : http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1002180 ; dataset : https://neurovault.org/collections/503/.

[4] D. J. Hand and V. Vinciotti, "Choosing k for two-class nearest neighbour classifiers with unbalanced classes," *Pattern Recognition Letters*, vol. 24, pp. 1555–1562, 2003.